PIECEWISE LINEARITIES IN NETWORK FLOWS AND SOLUTION APPROACHES

By

DUKWON KIM

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

1999

Dedicated to my family,
my wife (Jinkyung),
daughter (Alice), and son (Joshua)
with all my love from Jesus.

ACKNOWLEDGMENTS

First of all, I would like to thank my advisor, Professor Panos M. Pardalos, for his guidance, support, and much helpful advice.

I would also like to express my appreciation to the Department of Industrial and Systems Engineering at the University of Florida. In particular, the Center for Applied Optimization has provided me with an exceptionally rich environment for my study and research.

I would also like to thank my committee members, - Professors D. H. Hearn, B. D. Sivazlian, and Professor H. P. Benson - for devoting their time to the revision and enhancement of this thesis.

Special thanks to Professor H. Tuy from the Institute of Mathematics, Hanoi, Vietnam, for his valuable suggestions and encouragement during his visit to the Center for Applied Optimization.

Finally, I am most grateful to my family for their never-ending support and encouragement throughout all my endeavors.

TABLE OF CONTENTS

page

ACKNOWLEDGMENTS iii

LIST OF TABLES vi

LIST OF FIGURES vii

ABSTRACT ix

CHAPTERS

1 INTRODUCTION 1

- 1.1 Background 1
- 1.2 Network Flow Problem (NFP) 3
- 1.3 Nonconvex NFPs 5
- 1.4 Global Optimization 6
  - 1.4.1 Global Optimization Methods 7
  - 1.4.2 Computational Complexity 7
  - 1.4.3 Solution Approaches for Concave Minimization 8
- 1.5 Concave Minimization under Network Constraints 9

2 PIECEWISE LINEAR NETWORK FLOW PROBLEMS 11

- 2.1 Introduction 11
- 2.2 Problems and Formulations 13
  - 2.2.1 Convex Piecewise Linear Network Flow Problems 14
  - 2.2.2 Fixed Charge Network Flow Problems 15
  - 2.2.3 Concave Piecewise Linear Network Flow Problems 17
  - 2.2.4 Nonconvex Piecewise Linear Network Flow Problems 22
- 2.3 Applications and Algorithms 26
  - 2.3.1 Transportation Problems 26
  - 2.3.2 Location Problems 27
  - 2.3.3 Lot-sizing Problems 28
- 2.4 Concluding Remarks 30

3 FIXED CHARGE NETWORK FLOW PROBLEMS 31

3.1 Introduction 31
3.2 Problem Descriptions 33
3.3 Dynamic Slope Scaling Procedure 35
    3.3.1 Initial Solutions 38
    3.3.2 Updating Schemes 39
    3.3.3 Stopping Criterion 41
3.4 Computational Experiments 42
3.5 Local Search Procedure 49
    3.5.1 Greedy Adjacent Vertex Local Search Algorithm 50
    3.5.2 Numerical Results 50
3.6 Concluding Remarks 51

4 CONCAVE PIECEWISE LINEAR NETWORK FLOW PROBLEMS . . . 53

4.1 Introduction 53
4.2 Problems and Formulations 54
4.3 Trust Intervals and Solution Tendency 56
4.4 Algorithms 58
4.5 Computational Experiments 62
4.6 Concluding Remarks 66

5 NONCONVEX PIECEWISE LINEAR NETWORK FLOW PROBLEMS . 69

5.1 Introduction 69
5.2 The Problem and Formulations 70
5.3 Dynamic Domain Contraction Algorithm 74
5.4 Computational Experiments 78
5.5 Concluding Remarks 81

6 FUTURE RESEARCH 82

6.1 Application in Global Supply Chain Management 82
6.2 Piecewise Concave Nonlinear Arc Costs 86

REFERENCES 89

BIOGRAPHICAL SKETCH 95

LIST OF TABLES

Table page

3.1 Summary of computational results. 43

3.2 Computational results for large scale problems. 44

3.3 The Summary of $\mathsf{GAP}_{\mathrm{LP}}$, $\mathsf{RE}_{\mathrm{UB}}$ and $\mathsf{RE}_{\mathrm{ACP}}$. 47

3.4 f_{LP}, f_{UB} and f_{ACP} for large scale problems. 48

3.5 Summary of computational results of local search. 52

4.1 Computational results of DSSP in Approach I. 64

4.2 Computational results of DSSP in Approach II. 66

4.3 Computational results of DSSP in Approach III. 67

5.1 Summary of computational results for 200 problems. 80

LIST OF FIGURES

Figure page

2.1 An example of a concave piecewise linear arc cost function. 18

2.2 Arc separation procedure. 19

2.3 A staircase arc cost function. 23

2.4 A sawtooth arc cost function. 24

2.5 "All unit" quantity discounts. 29

3.1 Dynamic slope scaling factor $\bar{c}_j$. 36

3.2 The graphical illustration of relative errors. 45

3.3 Average CPU times: DSSP vs. B & B. 46

3.4 Average CPU times of DSSP for large scale problems. 46

3.5 A description of `fixnet6`. 49

3.6 A Greedy Adjacent Vertex Local Search (GAVLS) algorithm. 51

4.1 A graphical example of trust intervals and solution tendency 58

4.2 A graphical illustration of the initial solution. 60

4.3 A graphical illustration of the updating scheme. 61

4.4 A comparison of the average relative errors. 65

4.5 A comparison of the average CPU times. 65

5.1 An example of staircase arc cost functions. 71

5.2 An example of sawtooth arc cost functions. 72
5.3 An example of domain contraction. 76
5.4 An example of the reduced domain: Case I. 77
5.5 An example of the reduced domain: Case II. 78

6.1 The network representation of the Wagner-Whitin model. 85
6.2 A typical supply chain model. 86
6.3 A continuous piecewise concave arc cost function. 87
6.4 A discontinuous piecewise concave arc cost function. 88

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

PIECEWISE LINEARITIES IN NETWORK FLOWS AND SOLUTION APPROACHES

By

Dukwon Kim

August 1999

Chairman: Panos M. Pardalos
Major Department: Industrial and Systems Engineering

Network models are pervasive and arise in diverse application areas. Today, network flow optimization directly and indirectly influences high-level decision making in various industries, especially those related to transportation, production-inventory, distribution, supply chain, logistics, and communication network design problems.

In this thesis we are concerned with the two issues of piecewise linearity and reducibility in nonconvex network flow problems. The emphasis of this thesis focuses, firstly, on reducing each subclass of Piecewise Linear Network Problems (PLNFP) to a Fixed Charge Network Problem (FCNFP) and, secondly, on developing efficient solution approaches and algorithms for each class of problems.

The thesis consists of an introduction and a series of research papers. The common theme of these papers is the systematic use of the underlying network and the exploiting of piecewise linearities and separability of the cost structure to develop

efficient algorithms. We discuss the reducibility of PLNFPs into a fixed charge network flow model which is known to be $\mathcal{NP}$-hard and can be transformed to a 0-1 Mixed Integer Programming (MIP) problem.

Due to the exponentially increasing time and memory requirements for solving exactly such MIP problems, exact solution methods have typically failed to solve large-scale problems efficiently in most cases. Thus developing an effective and efficient heuristic approach is a main focus in this field. This motivates primary research on developing an efficient heuristic approach for the general FCNFP, called Dynamic Slope Scaling Procedure (DSSP). The extensive computational results show that the DSSP algorithm is highly efficient, effective and stable in its performance.

Based on these promising results, the DSSP approach is extended to solve more complicated problems such as pure concave PLNFPs and nonconvex (discontinuous) PLNFPs, incorporating Trust Intervals, Solution Tendency, and Domain Contraction techniques. Due to the flexibility of the algorithm structure, the DSSP can be easily implemented on various types of problems with practical cost structures including, economies of scale and nonconvex piecewise linear costs. As a practical application, a network solution approach for global supply chain management is briefly introduced for future work.

CHAPTER 1
INTRODUCTION

1.1 Background

Networks have become more and more apparent in our daily lives and network flow optimization directly and indirectly influences high-level decision making in various situations. Today it is almost impossible to name a field which has not benefited from network modeling or in which network optimization is not an important research component. It is not just in economics where network models have played a crucial role, but also in all aspects of industry and government planning.

Looking around us, we notice electric power networks bringing light and entertainment into our homes. Manufacturing and distribution networks give access to our lives necessities and consumer products. Every day we use telephone and satellite networks to communicate with each other across regional, national and international borders. Computer networks have had a great impact in Information Technology (IT), providing us with new ways of communicating and sharing information. In recent years, we have seen a dramatic explosion in IT. Today it is almost impossible to survive in the business world without having access to computer networks.

Networks have many special features including separability and unimodularity, and can be exploited by the problem solver in devising efficient solution algorithms. The network framework is often characterized by a diversity of problems that can be modeled and solved by network applications. In addition to the vast quantity of “real-world” applications, the special features of networks along with their very interesting and often very useful structure have captured many researchers’ attention.

The field of network theory has a long, rich tradition, tracing its roots back to the works of Kirchhof and other pioneers of electrical engineering who systematically analyzed electrical circuits. The earliest works describing an optimization problem involving network flows are generally attributed to Hitchcock [38], and the Russian Mathematician/economist Kantorovich [44]. Hitchcock developed one of the earliest network models describing what is now referred as the transportation problem. Kantorovich visualized the problem of the allocation of production levels to factories and the distribution of the resulting products to markets and thus developed one of the earliest network models within a practical framework.

These early works set the foundations of many of the key ideas behind network flow theory and provided the seeds of a general methodology for structuring and analyzing complex decision problems. Furthermore, they established the role of networks (graphs) as useful mathematical tools for representing many real-world problems. The first solution method with a theoretical guarantee of optimality is due to G.B. Dantzig. The result of his work in the form of the *Simplex Method* appeared in 1947 [18]. A great deal of the algorithmic development occurred during 1950s. However, since the theoretical solution capability was not immediately translated into computer procedures that could solve large-scale network problems in a reasonable length of time, the ability to solve network problems of real-world complexity remained undemonstrated. The identification of new problems which could be modeled and formulated as networks and the development of solution algorithms for these problems continued without a break. However, it was not until the late 1970s and early 1980s that this research resulted in the first reasonably fast computer codes. The computer software provided by this technology is currently in use. Due to the these developments, there has been a shift in research focus on a worldwide scale in the past two decades which has motivated many researchers to study networks and related problems [23, 68, 69].

1.2 Network Flow Problem (NFP)

The most fundamental of all network flow problems is the Minimum Cost Network Flow Problem (MCNFP). In fact, all other NFPs are special cases of MCNFP. Thus the statement of MCNFP can be given as follows:

> **MCNFP:** *Determine the least cost for shipping a commodity through a network such that the available supply at certain nodes can be used to satisfy the demand requirements at other nodes.*

Let $G = (N, A)$ be a directed graph consisting of a set N of m nodes and a set A of n arcs, then the mathematical formulation of MCNFP can be given as follows:

[MCNFP]

$$\min \quad f(x) = \sum_{(i,j)\in A} f_{ij}(x_{ij})$$

$$\text{s.t.} \quad \sum_{(k,i)\in A} x_{ki} - \sum_{(i,k)\in A} x_{ik} = b_i, \quad \forall i \in N$$

$$l_{ij} \leq x_{ij} \leq u_{ij}, \qquad \forall (i,j) \in A.$$

where x_{ij} is the amount of flow for each arc $(i,j) \in A$ and $f_{ij}(x_{ij})$ is the cost of moving x_{ij} units of flow through this arc. Although in most applications, this cost is more likely to be nonlinear, it is frequently assumed that the flow cost varies linearly with the amount of flow.

As mentioned earlier, the pure "linear" MCNFP is the most central of all network flow problems. Other problems like the Shortest Path Problem (SPP), Maximum Flow Problem (MFP), Transportation Problem (TP), Assignment Problem (AP), Circulation Problem (CP), Multicommodity Flow Problem (MCFP), etc. can be deduced from the basic formulation of MCNFP with relatively small modeling effort.

For example:

- The SPP, which is perhaps the simplest of all network flow problems, is a special case of MCNFP in which one wishes to determine the path that a single unit of flow travels between two prespecified nodes.
- In MFP, arc capacities are the only relevant parameters. The MFP is, in a sense, a complementary model to the SPP. In SPPs, one is concerned with situations in which flow incurs a cost but is not restricted by any capacities, while in contrast, the MFP incurs no costs but is restricted by the flow bounds.
- The formulation of TP differs from the basic MCNFP only by means of its bipartite network structure. Furthermore, in TP, it is assumed that the lower bound is zero and there are no capacity upper bounds on the arcs.
- The AP is a special case of TP where the supply at each source does not exceed one unit of flow.
- The CP is a MCNFP where all the nodes act as intermediate (transshipment) nodes i.e. $b_i = 0$, $\forall i \in N$.
- The MCFP models the situation where one is concerned with the flow of several commodities through the same underlying network. This differs from MCNFP where the model is only concerned with the flow of single commodity.

All of these problems and other related problems have been, and are still being extensively investigated by many researchers. A number of articles and books have been written, some identifying new problem classes with network structures, many developing and improving algorithms to solve these classes, and others surveying the work that has been done (see e.g. [1, 32, 43] and other references cited therein).

1.3 Nonconvex NFPs

In most network models considered in the literature, the involved costs are assumed to be linear and/or convex. In many of the applications encountered in practice however, this assumption fails to hold. In fact, many of the problems arising in production-inventory planning, economic lot sizing, location and network design exhibit economies of scale. The classical transportation model only involves linear or convex transportation costs. However, in many real-world applications such as listed above, the cost functions are concave (e.g. due to fixed charges and quantity discounts). An important class of decision problems deals with the design of optimal location and distribution systems. In theses problems, the objective is to minimize the total cost which consists of two parts: production costs, which usually exhibit economies of scale (or quantity discounts) and can be described by concave functions of the production quantities, and distribution-transportation costs, which can often be assumed to be linear. This composition of the costs is also typical for many other models of real-world applications in different areas (such as supply chain and logistics models). The latter problem, commonly known as the Production-Transportation Problem (PTP), has received a great deal of attention of many researchers due to the revival of supply chain management in recent years. A wide variety of problems encountered in applied fields such as operations research, economics, and engineering can be formulated as a Minimum Concave Cost Network Flow Problem (MCCNFP). This is another important and probably one of the most challenging problems of nonconvex network optimization.

In view of the nonconvexity of the objective function, these problems belong to the realm of *global optimization*. In the following sections, we briefly introduce the area of global optimization and then turn to the issue of global optimization under network constraints.

1.4 Global Optimization

Global optimization is a relatively new field. It was initiated in the mid 1960s, mainly through the works of Hoang Tuy. However, since the primary works of Tuy [77], global optimization has become one of the most active research areas.

Nonconvex Optimization appears in a huge variety of forms and problem classes, including nonlinear, stochastic and combinatorial programming, multiobjective optimization, control, games and approximation theory. Certain important classes of nonlinear optimization have the property that all local optima are global optima. Thus, for these classes, standard optimization procedures which yield local optimal solutions also solve the global optimization problem. In general, however, when considering a nonconvex problem, one faces the difficulty of multiextremality. This means that the problem possesses several local optima where the corresponding function values may differ substantially. The branch of optimization which is concerned with the design of algorithms that distinguish between these local optima in order to identify the best one possible is known as *global optimization.* The general global optimization problem is defined as follows [42].

> *Given a nonempty, closed set $\mathcal{D} \subset I\!R^n$ and a continuous function $f : \Omega \mapsto I\!R$, where $\Omega \subset I\!R^n$ is a suitable set containing $\mathcal{D}$, find at least one point $x^* \in \mathcal{D}$ satisfying $f(x^*) \leq f(x)$ for all $x \in \mathcal{D}$.*

In view of this definition, any point $x^* \in \mathcal{D}$ such that $f(x^*) \leq f(x)$, $\forall x \in \mathcal{D}$ is called a *global minimizer.* This is in contrast to a *local minimizer* i.e., any point $\bar{x} \in \mathcal{D}$ for which there exists a neighborhood $\aleph$ of $\bar{x}$ satisfying $f(\bar{x}) \leq f(x)$, $\forall x \in \aleph \cap \mathcal{D}$. Global optimization thus aims at determining not just a local optima but the *best* among all local optimizers in the feasible region $\mathcal{D}$.

1.4.1 Global Optimization Methods

Due to their multiextremality, the use of standard local optimization techniques is normally insufficient for solving global optimization problems. For this reason, more sophisticated methods designed for global optimization are important from a practical point of view. This need has been widely recognized and a variety of global optimization algorithms have been developed over the past few years. However, particular approaches suggested for solving the global optimization problem are often of a very different nature. In spite of this, there exist some characteristics that are common for all methods. In general, a method utilizing all *a priori* information about the problem would outperform a method utilizing less information. Therefore, the design of a method will strongly depend on the *a priori* information available. Detailed and comprehensive surveys of global optimization methods are given in [3, 22, 41, 76].

1.4.2 Computational Complexity

A major difficulty of global optimization problems is that many local optima can occur. One conspicuous limitation of local optimization methods is that they become trapped at a local optimum, or more generally, at a stationary point for which there is not even any guarantee for local optimality [40]. In addition to the multiextremality of the problem, this causes the failure of standard local nonlinear programming procedures in solving global optimization problems. Therefore, global optimization techniques are generally more complex, and consequently more computationally expensive, than conventional local nonlinear optimization methods.

From the computational complexity point of view, Garey and Johnson [29] and Mangasarian and Shiau [57] showed that even the special case of minimizing a quadratic concave function over the unit hypercube is an $\mathcal{NP}$-hard problem. Pardalos and Vavasis [67] proved that introducing a single nonconvex variable into the objective function makes the problem $\mathcal{NP}$-hard. Pardalos and Schnitger [65] also proved that even the problem of checking whether a feasible point is a local (global) optima in quadratic programming is $\mathcal{NP}$-hard. Thus it is evident that global optimization problems are in general $\mathcal{NP}$-hard. In particular, the complexity of general Minimum Concave Cost Network Flow Problems (MCCNFP) was investigated by Guisewite and Pardalos [32, 35].

This suggests that one should look for specially structured global optimization problems which can be efficiently solved by properly adapted methods. One important class of such problems, involving a network structure, has been considered extensively in the literature.

1.4.3 Solution Approaches for Concave Minimization

Most deterministic approaches for global concave programming use vertex enumeration techniques, cutting plane methods, branch and bound or different combinations of these techniques. There are, however, specific methods that have been proposed for treating specially structured problems. This special structure may appear in the objective function in the form of *separability* or *piecewise linearity.* It is also possible that the feasible set has a special structure (e.g. hypercube, network constraints, etc.) that has been exploited in the development of solution algorithms.

A bibliographic survey which summarizes the main ideas of concave programming is presented by Pardalos and Rosen [63]. This survey also includes results concerned with large scale global concave minimization and bilinear programming. The most recent review article is due to Benson [9] which gives a comprehensive survey of the development of the subject up to 1993.

1.5 Concave Minimization under Network Constraints

H.Tuy [77] was among the first authors to consider the development of methods for solving concave minimization problems under linear constraints. His idea was based on the combination of cuts (additional linear constraints) to exclude part of the feasible domain with conical partitioning of the region.

One of the most important classes in concave minimization is the problem of minimizing a concave function under network constraints. This class of global optimization problems has been studied extensively. A wide range of results for the Minimum Concave Cost Network Flow Problem (MCCNFP), including related applications, complexity issues and solution techniques is given in [32]. A very general problem falling into the category of network constrained concave minimization is the MCCNFP. This problem is defined as that of finding a minimum-cost flow in a network where the flow cost in each arc is additive and concave. Many problems encountered in applied fields such as production-inventory, capacity planning, economic lot sizing, transportation, warehouse location, and communication network design, that exhibit *economies of scale* and/or *quantity discounts* can be modeled as an MCCNFP. This problem, like most nonconvex optimization problems, belongs to the realm of $\mathcal{NP}$-hard problems [29]. Hence, all available algorithms for the general problems run in exponential time. In view of its importance in both global and combinatorial optimization, MCCNFP has been investigated by many researchers. Although the MCCNFP can be solved by an exhaustive search of all the extreme points of feasible domain (usually polyhedral set), such a solution approach is impractical for all but small-size problems. The first branch and bound algorithm for MCCNFP was presented in [70]. This idea is based on the decomposition of the original problem into a sequence of convex piecewise linear problems which are solved by the *out-of-kilter* algorithm. With this algorithm, it is also possible to handle arc capacities. Another branch and bound algorithm for MCCNFP is given in [25]. This method is based on the equivalence of the general network flow problem to a network flow problem on a bipartite network of a special form. An optimal flow which is a spanning tree, is found by an implicit enumeration of the set of extremal flows in that network. The single source multiple destination MCCNFP is considered in [28]. The global optimum is sought using a branch and bound approach, in which the enumeration scheme is based on a characterization of the optimal solution set, while linear relaxations of the original problem provide lower bounds. Other algorithms for the single-source uncapacitated version of MCCNFP are discussed by Guisewite and Pardalos [33, 34].

CHAPTER 2
PIECEWISE LINEAR NETWORK FLOW PROBLEMS

2.1 Introduction

In this chapter we study a subclass of minimum cost network flow problems in which the cost is a separable piecewise linear function of the total flow along the arcs. The problems can be categorized according to the structure of cost functions, such as convex, concave, and nonconvex (*indefinite*) Piecewise Linear Network Flow Problems (PLNFP). General minimum linear (*including convex*) cost network flow problems are well known and well solved problems. Unfortunately, however, these cost structures are too simple to model many real world situations. In order to improve the models, concave or nonconvex piecewise linear cost functions have been introduced into many application areas where the cost function exhibits strong economies of scale. This allows fixed costs to appear at several levels of activity, and also allows the linear cost coefficients to vary between different intervals.

Some areas of applications are transportation planning, production-inventory control, economic lot-sizing problems, facility location and capacity expansion planning, and communication network design. Clearly, the transportation problem with a fixed charge is the most popular application area in the literature.

Another problem that can be modeled as a concave-cost network flow problem is the design of a local communication network. In this problem, each customer in a region must be connected to a central office. Instead of using a dedicated line between each customer and the central office, concentrators are used at intermediate locations in order to take advantage of economies of scale (*or piecewise linearity*) in communication costs. The goal is, therefore, to select the locations of concentrators and their capacities and to connect the customers to the central office via the concentrators in order to minimize total number of concentrators and communication costs.

Past research in this area has primarily focused on solving the general concave cost network problem rather than the special case of piecewise linear cost function that we are considering in this paper. The methods used to solve the general concave problem may not be as effective as algorithms that exploits the special structure of piecewise linear cost functions. Those methods include branch-and-bound and dynamic programming. Heuristic approaches have also been used in many different schemes. A recent survey in this area can be found in [32].

The chapter provides a brief description and formulations of the problem in each category. As we can see in the following section, concave and nonconvex PLNFP are formulated as a Fixed Charge Network Flow Problem (FCNFP) which have been well studied in the sense of global optimization.

The most interesting fact, here, is that the nonconvex (*discontinuous*) PLNFP can be transformed to an equivalent FCNFP model with additional variables and constraints. As mentioned above, the FCNFP can be solved more efficiently than the original nonconvex PLNFP itself. However, extreme point solutions of the resulting FCNFP may not be the extreme points of the feasible region in the original nonconvex PLNFP model since the feasible region is modified with additional constraints. The discussion of this transformation is given in Section 2.2.4 regarding two typical nonconvex piecewise linear cost functions, which are *staircase* and *sawtooth* arc costs.

2.2 Problems and Formulations

As a special subclass of minimum cost network flow problems, general piecewise linear network problems can be classified according to the type of piecewise cost functions. Using general network flow constraints, PLNFP can be stated as follows:

Given a directed graph $G = (N, A)$ consisting of a set N of m nodes and a set A of n arcs, then solve

[PLNFP]

$$\min \ f(x) = \sum_{(i,j)\in A} f_{ij}(x_{ij})$$

subject to

$$\sum_{(k,i)\in A} x_{ki} - \sum_{(i,k)\in A} x_{ik} = b_i, \ \ \forall i \in N \tag{2.1}$$

$$0 \le x_{ij} \le u_{ij}, \ \ \forall (i,j) \in A \tag{2.2}$$

where f is separable and each f_{ij} is piecewise linear. For instance, the arc cost $f_{ij}(x_{ij})$ can be defined as follows:

$$f_{ij}(x_{ij}) = \begin{cases} c_{ij}^1 x_{ij} + s_{ij}^1 & \text{if } 0 \le x_{ij} < \lambda_{ij}^1 \\ c_{ij}^2 x_{ij} + s_{ij}^2 & \text{if } \lambda_{ij}^1 \le x_{ij} < \lambda_{ij}^2 \\ \vdots & \vdots \\ c_{ij}^{r_{ij}} x_{ij} + s_{ij}^{r_{ij}} & \text{if } \lambda_{ij}^{r_{ij}-1} \le x_{ij} \le \lambda_{ij}^{r_{ij}} (= u_{ij}), \end{cases} \tag{2.3}$$

where λ_{ij}^k for $k = 1$ to $r_{ij} - 1$ are breakpoints in the given interval $[0,\ u_{ij}]$ and r_{ij} is the number of linear pieces in each arc cost function f_{ij}. The constraints in (2.1) are called the conservation of flow equations. The constraints in (2.2) are called capacity constraints on the arc flows. The problem is uncapacitated if $u_{ij} = \infty$, $\forall (i,j) \in A$.

Three specific classes in PLNFP can be identified based on the arc costs, f_{ij} as: 1) Convex PLNFP, 2) Concave PLNFP, and 3) Nonconvex PLNFP[1].

---

[1] In some cases, it is called *discontinuous* PLNFP, since it usually results from a set of discontinuities in the arc cost functions.

Since the Fixed Charge Network Flow Problem (FCNFP) has a very close relation to the PLNFP, it is important to understand the special structure of FCNFP to solve PLNFP. Due to the global optimality property of concave minimization [41, 64], global solutions can be obtained at extreme points of feasible regions for the cases of FCNFP and concave PLNFP. A recent survey on minimum concave-cost network flow problems can be found in [32]. These problems including FCNFP are discussed in the following sections.

2.2.1 Convex Piecewise Linear Network Flow Problems

Suppose that constraints are all linear, and the cost function to be minimized is separable and piecewise linear. Then the proportionality assumption is violated for the cost function. However, if the piecewise linear function is convex, the problem can still be formulated as a Linear Programming (LP) problem [60].

Let $f_{ij}(x_{ij})$ denote the contribution of x_{ij} to the separable objective function. Suppose that there are $r_{ij}-1$ breakpoints at which $f_{ij}(x_{ij})$ changes slope, such that

$$0 = \lambda_{ij}^{0} < \lambda_{ij}^{1} < \lambda_{ij}^{2} < \ldots < \lambda_{ij}^{r_{ij}-1} < \lambda_{ij}^{r_{ij}} = u_{ij}.$$

Let the slope in the subinterval $\lambda_{ij}^{k-1} \leq x_{ij} \leq \lambda_{ij}^{k}$ be c_{ij}^{k} for $k = 1$ to r_{ij}, and let y_{ij}^{k} be the portion of x_{ij} lying in the kth subinterval, λ_{ij}^{k-1} to λ_{ij}^{k}, (i.e., y_{ij}^{k} is the length of the overlap of the interval 0 to x_{ij} with the subinterval λ_{ij}^{k-1} to λ_{ij}^{k}), $k = 1$ to r_{ij}. When defined in this manner, the new variables $y_{ij}^{1}, y_{ij}^{2}, \ldots, y_{ij}^{r_{ij}}$ partition x_{ij} such that

$$x_{ij} = y_{ij}^{1} + y_{ij}^{2} + \ldots + y_{ij}^{r_{ij}}. \tag{2.4}$$

These variables are subject to the constraints:

$$\begin{array}{rcl} 0 \leq & y_{ij}^1 & \leq \lambda_{ij}^1 \\ 0 \leq & y_{ij}^2 & \leq \lambda_{ij}^2 - \lambda_{ij}^1 \\ & \vdots & \\ 0 \leq & y_{ij}^{r_{ij}-1} & \leq \lambda_{ij}^{r_{ij}-1} - \lambda_{ij}^{r_{ij}-2} \\ 0 \leq & y_{ij}^{r_{ij}} & \leq \lambda_{ij}^{r_{ij}} - \lambda_{ij}^{r_{ij}-1} \end{array} \tag{2.5}$$

and

$$\text{for every } t, \text{ if } y_{ij}^t > 0, \text{ then each of } y_{ij}^k \text{ is equal to} \tag{2.6}$$

$$\text{its upper bound } \lambda_{ij}^k - \lambda_{ij}^{k-1}, \ \forall\, k < t.$$

Defining the new variables as shown above, it is clear that $f_{ij}(x_{ij})$ is equal to $c_{ij}^1 y_{ij}^1 + \ldots + c_{ij}^{r_{ij}} y_{ij}^{r_{ij}}$. If the original separable piecewise linear objective function to be minimized is continuous and convex so that it holds the following *increasing* conditions

$$c_{ij}^1 < c_{ij}^2 < \ldots < c_{ij}^{r_{ij}}, \tag{2.7}$$

constraints on the new variables of the type (2.6) can be ignored in the transformed model. Since convex PLNFP is reformulated as an LP using the above technique and has specially structured network constraints, it can be solved efficiently in polynomial time. If f_{ij} is not continuous or the slopes do not satisfy the condition (2.7), then the constraints (2.6) must be specifically included in the model. Since these constraints are not linear, the transformed model is no longer an LP.

2.2.2 Fixed Charge Network Flow Problems

Due to the similarity of its structure with the piecewise linear case, the Fixed Charge Network Flow Problem (FCNFP) has close relations to PLNFPs. It is very important and useful to investigate the structure of FCNFP to study the PLNFP.

The FCNFP is a special case of Minimum Concave Cost Network Flow Problems (MCCNFP) [41], whose arc cost function has a discontinuity at the origin. The arc cost function $f_{ij}(x_{ij})$ of the FCNFP has a form

$$f_{ij}(x_{ij}) = \begin{cases} 0 & \text{if } x_{ij} = 0 \\ s_{ij} + c_{ij}x_{ij} & \text{if } x_{ij} > 0, \end{cases} \tag{2.8}$$

where $s_{ij} \geq 0$ is a fixed cost for arc $(i, j) \in A$.

In many practical problems, the cost of an activity is the sum of a fixed cost and a cost proportional to the level of the activity. FCNFP is obtained by imposing a fixed cost of $s_{ij} \geq 0$ if there is positive flow on arc (i, j) and a variable cost c_{ij}. Due to the discontinuity of f_{ij}, the problem can be transformed into a 0-1 Mixed Integer Programming (MIP) problem by introducing n binary variables, indicating whether the corresponding activity is being carried out or not. Assuming $s_{ij} > 0$, f_{ij} can be replaced by

$$f_{ij} = c_{ij}x_{ij} + s_{ij}y_{ij}$$

with

$$x_{ij} \geq 0 \text{ and } y_{ij} = \begin{cases} 0 & \text{if } x_{ij} = 0 \\ 1 & \text{if } x_{ij} > 0. \end{cases} \tag{2.9}$$

The above condition (2.9) can be incorporated into the capacity constraints to yield

$$0 \leq x_{ij} \leq u_{ij}y_{ij}, \quad y_{ij} \in \{0, 1\}.$$

Hence we obtain the following formulation of the fixed charge network flow problem:

[FCNFP]$_{\mathbf{MIP}}$

$$\min \sum_{(i,j)\in A} (c_{ij}x_{ij} + s_{ij}y_{ij})$$

subject to

$$Ex = b \tag{2.10}$$

$$0 \leq x_{ij} \leq u_{ij} y_{ij}, \qquad (i,j) \in A \tag{2.11}$$

$$y_{ij} \in \{0,1\}, \qquad (i,j) \in A, \tag{2.12}$$

where E is an $m \times n$ node-arc incidence matrix and b is an m-dimensional column vector. This $[\mathbf{FCNFP}]_{\mathbf{MIP}}$ can be solved by using any type of classical branch and bound algorithms that use LP relaxations [61]. These LP relaxations can be solved efficiently by existing linear network algorithms exploiting the special structure of their feasible domain [41].

As we can see in later sections, many concave and nonconvex PLNFPs can be reduced to a FCNFP model by introducing new variables and modifying problem structures. It is noticed that FCNFP models reduced from original PLNFPs can be also transformed to a 0-1 mixed integer programming problem. Consequently, the size of the resulting model grows very fast even if the original PLNFP is just of medium size. This stimulates the reason that many researchers have developed new efficient schemes to improve their exact methods (*especially the branch and bound method)*.

Indeed, the computational effort and memory requirement to solve large-scale FCNFP models have been gradually reduced in various application areas. Yet, since there is a limitation for improving exact solution methods to solve the problem in practical sense, developing a effective approximate solution method is still in need.

2.2.3 Concave Piecewise Linear Network Flow Problems

As different from the convex case in Section 2.2.1, concave PLNFPs are more difficult to solve since we cannot use the same technique in the convex case to reduce the problem into an LP. However, a concave PLNFP can be transformed to a fixed charge network problem in an extended network. The size of the extended network

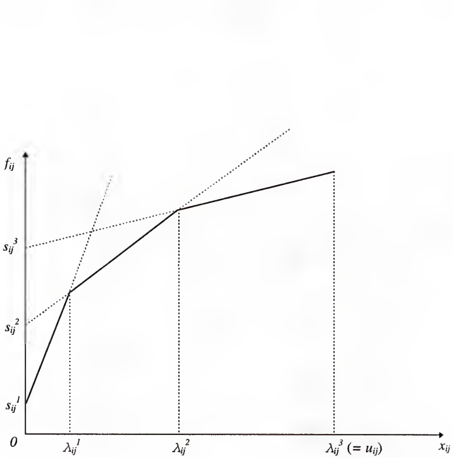

Figure 2.1. An example of a concave piecewise linear arc cost function.

depends on the number of linear pieces in each arc cost function. An *Arc Separation Procedure* (ASP) is required to solve the problem in this way and ASP can be valid due to the concavity of arc cost functions.

Let us consider an arc $(i, j) \in A$ and its arc cost f_{ij}, and suppose f_{ij} has r_{ij} linear pieces as defined in (2.3). Then arc (i, j) can be separated into r_{ij} arcs between nodes i and j for $(i, j) \in A$. Each separated arc $(i, j)^k$ for $k = 1$ to r_{ij} has a fixed charge cost function f_{ij}^k (*see* Figure 2.1) defined by

$$f_{ij}^k(x_{ij}^k) = \begin{cases} 0 & \text{if } x_{ij}^k = 0 \\ s_{ij}^k + c_{ij}^k x_{ij}^k & \text{if } x_{ij}^k > 0, \end{cases}$$

where x_{ij}^k is a flow for the separated arc $(i, j)^k$. This extended network is denoted by

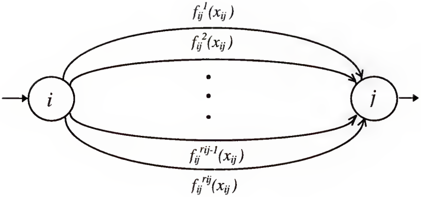

Figure 2.2. Arc separation procedure.

$G_e(N, A_e)$ where the number of arcs $|A_e|$ is given by

$$n_e = |A_e| = \sum_{(i,j)\in A} r_{ij}.$$

After the ASP modification shown in Figure 2.2, the original concave piecewise linear objective function can be expressed as a sum of fixed charge arc cost functions as follows:

$$\begin{aligned} f(x) &= \sum_{(i,j)\in A} f_{ij}(x_{ij}) \\ &= \sum_{(i,j)\in A} \sum_{k=1}^{r_{ij}} f_{ij}^{k}(x_{ij}^{k}) \end{aligned} \tag{2.13}$$

It is easy to see that the equality in (2.13) can not be true in general cases without a set of constraints to restrict a domain for each separated arc cost function. However, due to the following property from the concavity of arc cost functions:

$$c_{ij}^{1} > c_{ij}^{2} > \ldots > c_{ij}^{r_{ij}} > 0, \tag{2.14}$$

the equality holds for true in this case. More precisely, $f_{ij}(x_{ij})$ is equal to at most one of arc cost among all separated arc costs between node i and j at the optimality of minimization problems. This argument can be generalized as the following theorem.

Theorem 2.1 *Given an extended network described above, if a positive flow* x^*_{ij} *is optimal for a minimum concave PLNFP and if* $\lambda^{q-1}_{ij} < x^*_{ij} \leq \lambda^q_{ij}$ *for* $1 \leq q \leq r_{ij}$*, then it takes only one arc,* $(i,j)^q$ *(i.e. q-th arc) among all separated arcs between node i and j,* $(i,j)^k$ *for* $k = 1$ *to* r_{ij}*.*

Proof: The same notation as the above is used in the proof.

Let us consider an arc $(i,j) \in A$ with r_{ij} pieces of concave piecewise linear cost function f_{ij}. And denote that c^k_{ij} and s^k_{ij} are slope and fixed charge of the kth piece of linear function for $k = 1, 2, \ldots, r_{ij}$, respectively. Notice that these slopes hold the property in (2.14). After ASP, there are r_{ij} separated arcs the flow can take. So, let us denote that $flow^k$ is the optimal flow of the kth separated arc $(i,j)^k$ for $k = 1, 2, \ldots, r_{ij}$.

Firstly, it is clear that if x^*_{ij} less than equal to the first breakpoint λ^1_{ij}, then the optimal flow can not be split (i.e. $flow^1 = x^*_{ij}$) since the cost of arc $(i,j)^1$, $f^1_{ij} \leq f^k_{ij}$, $\forall\ x_{ij} \leq \lambda^1_{ij}$ for $k = 2, 3, \ldots, r_{ij}$.

Secondly, consider the following two cases when $\lambda^p_{ij} \leq \lambda^{q-1}_{ij} < x^*_{ij} \leq \lambda^q_{ij}$, and suppose $flow^p$ and $flow^q$ are optimal flows in the extended network problem.

- **Case I :** $flow^p = \varepsilon$ and $flow^q = x^*_{ij} - \varepsilon > 0$ for any positive ε.

  Using a contradiction, suppose that the optimal flow is split into two arcs $(i,j)^p$ and $(i,j)^q$, $1 \leq p < q \leq r_{ij}$. Now, let us evaluate the difference of cost function values in the original PLNFP with x^*_{ij} and the extended network problem with $flow^p$ and $flow^q$:

$$\begin{aligned} & [f^p_{ij}(\varepsilon) + f^q_{ij}(x^*_{ij} - \varepsilon)] - f_{ij}(x^*_{ij}) \\ = \; & [s^p_{ij} + c^p_{ij}\varepsilon + s^q_{ij} + c^q_{ij}(x^*_{ij} - \varepsilon)] - (s^q_{ij} + c^q_{ij}x^*_{ij}) \\ = \; & s^q_{ij} + \varepsilon(c^p_{ij} - c^q_{ij}) \\ > \; & 0. \end{aligned}$$

The inequality in the last term above is true since $s^q_{ij} \geq$ and $c^q_{ij} < c^p_{ij}$. Thus, it contradicts to the optimality of $flow^p$ and $flow^q$.

- **Case II :** $flow^p = \varepsilon = x^*_{ij}$ and $flow^q = 0$.

  Again, using a contradiction, let us evaluate the difference of cost function value between node i and j:

  $$\begin{aligned} & [f^p_{ij}(\varepsilon) + f^q_{ij}(0)] - f_{ij}(x^*_{ij}) \\ = \; & (s^p_{ij} + c^p_{ij}\varepsilon) - (s^q_{ij} + c^q_{ij}x^*_{ij}) \\ = \; & (s^p_{ij} + c^p_{ij}x^*_{ij}) - (s^q_{ij} + c^q_{ij}x^*_{ij}) \\ > \; & 0. \end{aligned}$$

  With the same reason in **Case I**, we have a contradiction to the optimality of $flow^p$ and $flow^q$.

Therefore, $flow^q = x^*_{ij}$ and the proof is completed. $\square$

Based on the theorem above, the original concave PLNFP can be reduced to a FCNFP with the objective function given in (2.13) and some extended network constraints corresponding to the constraints in (2.1). The resulting formulation of a concave PLNFP is shown as follows:

[PLNFP]$_{\mathbf{FC}}$

$$\min \quad f(x) = \sum_{(i,j)\in A} \sum_{k=1}^{r_{ij}} f^k_{ij}(x^k_{ij})$$

subject to

$$\sum_{(l,i)\in A} \sum_{k=1}^{r_{ij}} x^k_{li} - \sum_{(i,l)\in A} \sum_{k=1}^{r_{ij}} x^k_{il} = b_i, \quad \forall i \in N \tag{2.15}$$

$$\begin{aligned} \sum_{k=1}^{r_{ij}} x_{ij}^k = x_{ij}, &\qquad \forall (i,j) \in A \\ x_{ij}^k \geq 0, &\qquad \forall (i,j) \in A,\ k = 1, 2, \ldots, r_{ij} \\ 0 \leq x_{ij} \leq u_{ij}, &\qquad \forall (i,j) \in A. \end{aligned}$$

In the formulation, it is clear that any other set of constraints is not necessary to specify lower and upper bounds for separated arcs $(i,j)^k \in A_e,\ \forall\, i,j \in N$ due to the above theorem. As a result, the solution of concave PLNFP can be found by solving the fixed charge network problem formulated as above. Thus, developing an efficient algorithm to solve a FCNFP (*exactly or approximately*) is a key to solve PLNFPs in this approach.

2.2.4 Nonconvex Piecewise Linear Network Flow Problems

Lastly, we consider a PLNFP with nonconvex (*indefinite*) arc cost functions, which is the most difficult case in this class. One of the major difficulty to find exact solutions for nonconvex PLNFPs is originated from the structure involving discontinuities and nonsmoothness in their arc cost function. Obviously such cost functions are neither convex nor concave, and possibly have a finite set of discontinuities. However, due to the nature of real world applications of the model, we focus on two certain types of arc cost functions in nonconvex PLNFP models, so-called *staircase* [11, 39, 52] and *sawtooth* [52] arc cost functions, respectively.

Both arc cost functions have a very similar structure in overall shape, however, they have a different aspect at breakpoints. It can be described in mathematical form as follows:

- "Staircase" arc cost function: *see* Figure 2.3.

$$f_{ij}^{k-1}(\lambda_{ij}^{k-1}) < f_{ij}^k(\lambda_{ij}^{k-1} + \varepsilon), \quad \text{for any } \varepsilon > 0 \text{ and } k = 2, 3, \ldots, r_{ij}.$$

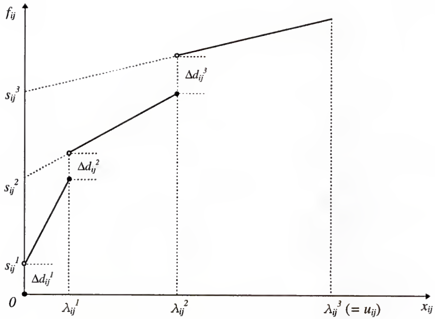

Figure 2.3. A staircase arc cost function.

- "Sawtooth" arc cost function: *see* Figure 2.4.

$$f_{ij}^{k-1}(\lambda_{ij}^{k-1} - \varepsilon) > f_{ij}^{k}(\lambda_{ij}^{k-1}), \quad \text{for any } \varepsilon > 0 \text{ and } k = 2, 3, \ldots, r_{ij}.$$

Moreover, it is assumed that the property of slopes shown in (2.14) is still valid since it is a very general phenomenon in real applications. Note that extreme point solutions are not guaranteed in this cases since objective functions are no longer concave.

Now, we introduce an equivalent formulation of the problem, $[\mathbf{FCNFP}]_{\mathbf{MIP}}$ with some additional parameters and binary variables. Let us define the size of interval between adjacent breakpoints as

$$\Delta\lambda_{ij}^k = \lambda_{ij}^k - \lambda_{ij}^{k-1}, \quad \forall (i,j) \in A, \quad k = 1, 2, \ldots, r_{ij}, \tag{2.16}$$

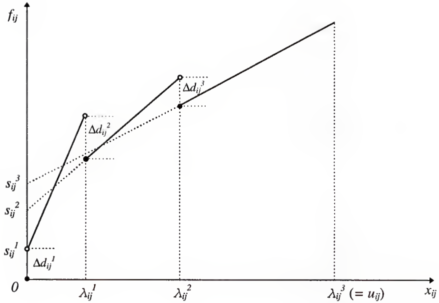

Figure 2.4. A sawtooth arc cost function.

and define the gap of function values at each breakpoint in arc cost functions as

$$\begin{aligned}\Delta d_{ij}^{k} &= (c_{ij}^{k}\lambda_{ij}^{k-1} + s_{ij}^{k}) - (c_{ij}^{k-1}\lambda_{ij}^{k-1} + s_{ij}^{k-1}) \\ &= s_{ij}^{k} - s_{ij}^{k-1} + (c_{ij}^{k} - c_{ij}^{k-1})\lambda_{ij}^{k-1}, \quad \forall (i,j) \in A,\ k = 1, 2, \ldots, r_{ij},\end{aligned} \tag{2.17}$$

where $s_{ij}^{0} = 0$ and $c_{ij}^{0} = 0$ (also clearly $\Delta d_{ij}^{1} = s_{ij}^{1}$). We now let x_{ij}^{k} be the part of x_{ij} that lies within level k (i.e. kth subinterval), in the following sense,

$$x_{ij}^{k} = \begin{cases} 0 & \text{if } x_{ij} \leq \lambda_{ij}^{k-1}, \\ x_{ij} - \lambda_{ij}^{k-1} & \text{if } \lambda_{ij}^{k-1} < x_{ij} \leq \lambda_{ij}^{k}, \\ \Delta\lambda_{ij}^{k} & \text{if } x_{ij} \geq \lambda_{ij}^{k}, \end{cases} \tag{2.18}$$

and we obtain the following equation for substitution into **[FCNFP]$_{\mathbf{MIP}}$** model,

$$x_{ij} = \sum_{k=1}^{r_{ij}} x_{ij}^{k}. \tag{2.19}$$

We then introduce new binary variables defined by

$$y_{ij}^{k} = \begin{cases} 1 & \text{if } \lambda_{ij}^{k-1} < x_{ij} \\ 0 & \text{otherwise.} \end{cases} \tag{2.20}$$

Using (2.16) - (2.20), the nonconvex PLNFP under consideration can be formulated as an MIP version of FCNFP as follows:

[NPLNFP]$_{\mathbf{MIP}}$

$$\min \sum_{(i,j)\in A} \sum_{k=1}^{r_{ij}} (c_{ij}^k x_{ij}^k + \Delta d_{ij}^k y_{ij}^k)$$

subject to constraints in (2.1) and

$$x_{ij} = \textstyle\sum_{k=1}^{r_{ij}} x_{ij}^k \quad \forall (i,j) \tag{2.21}$$

$$x_{ij}^k \leq \Delta\lambda_{ij}^k y_{ij}^k \quad \forall (i,j),\ k = 1,2,\ldots,r_{ij} \tag{2.22}$$

$$x_{ij}^{k-1} \geq \Delta\lambda_{ij}^{k-1} y_{ij}^k \quad \forall (i,j),\ k = 2,3,\ldots,r_{ij} \tag{2.23}$$

$$x_{ij}^k \geq 0 \quad \forall (i,j),\ k = 1,2,\ldots,r_{ij} \tag{2.24}$$

$$y_{ij}^k \in \{0,1\} \quad \forall (i,j),\ k = 1,2,\ldots,r_{ij}. \tag{2.25}$$

It is noticed that combining one constraint from (2.22) and one from (2.23) yields $\Delta\lambda_{ij}^{k-1} y_{ij}^k \leq x_{ij}^{k-1} \leq \Delta\lambda_{ij}^{k-1} y_{ij}^{k-1}$, which implies $y_{ij}^k \leq y_{ij}^{k-1}$, $\forall (i,j)$, $k > 1$.

There is another approach to formulate the problem as a concave Minimum Cost Network Flow Problem (MCNFP) model. In [52], Lamar described an equivalent formulation of the MCNFP with general nonlinear arc costs (including the problem considered in this section) as a concave MCNFP on an extended network. The equivalence between the problems is based on converting each arc with an arbitrary cost function in the original problem into an arc with a concave piecewise linear cost function in series with a set of parallel arcs, each with a linear arc cost function (*see* [52] for details). Thus, the resulting problem is a concave MCNFP, which is different from the FCNFP formulation model shown above.

2.3 Applications and Algorithms

Piecewise linear network models have a number of applications in various areas such as transportation problems, location problems, distribution problems, communication network design problems, and economic lot-sizing problems.

Due to the structure of objective functions (especially in concave and nonconvex PLNFP models), many real-world situations listed above can be modeled as a PLNFP or its variant. One of the interesting applications is economic lot-sizing problems which usually involve fixed costs (setup costs) in most cases. It is known that the dynamic programming approach proposed by Chen *et al.* [14, 15] is one of the most efficient methods. There are major applications in the following three fields which have been studied extensively in the literature.

2.3.1 Transportation Problems

The first major field is transportation-related problems with concave cost functions including fixed charge cost functions. The concave cost functions in this field are usually assumed to be piecewise linear in many cases. A number of algorithms developed with different schemes and their computational results have been reported. These can be found in a limited list of references [4, 5, 6, 13, 21, 31, 45, 46, 51, 53, 62, 70, 71].

The category of exact solution approaches contains diverse techniques based on extreme point ranking, branch and bound, and dynamic programming methods. Since the problems can be formulated as MIP models, the branch and bound approach with various branching schemes has been a major interest in the literature.

Murty [59] introduced an extreme point ranking method for solving fixed charge problems. McKeown [58] extended Murty's method to avoid some degeneracy of the problem. Recently, Pardalos [66] discussed a range of enumerative techniques based on vertex enumeration or extreme point ranking.

Rech and Barton [70] investigated a nonconvex transportation algorithm using the branch and bound approach to problems with piecewise linear cost functions. These functions are approximated by a convex envelope and solved by using out-of-kilter method. Bornstein and Rust [11] specialized this approach to the problem with staircase cost function, using successive linearizations of the objective function. Thach [74] proposed a method for decomposing the problem with a staircase structure into a sequence of much smaller subproblems.

Lamar [53] developed a branch and bound approach for cases of capacitated MC-CNFP in which the costs consist of piecewise linear segments. The problem is formulated as an MIP, with the branching variables determining which linear cost region an arc flow falls into.

Recently, Kim and Pardalos [48] developed a heuristic procedure for solving general FCNFP without formulating it as an MIP. The procedure is consist of solving a series of LPs to update slopes and searching extreme points of the convex feasible region with the updated slopes. This approach provides a potential possibility of parallel implementation with different initial solutions to improve the quality of solutions. Some heuristic approaches can be found in [16, 21, 46, 51].

2.3.2 Location Problems

Another major application area is to solve location problems. Since the problem in this field is to locate facilities and determine the size of facilities to minimize total cost [26], it naturally involves fixed costs and/or piecewise linear costs. Since solution methods in this field have used network formulation in many cases, they are quite similar to those for solving FCNFP or PLNFP [2, 19, 24, 47, 72]. However, exploiting their problem structures, there are some Lagrangean approaches [7, 30] to the problems.

Recently, Holmberg [39] proposed a decomposition and linearization approach for solving the facility location problem with staircase costs. A comparison of heuristic and relaxation approaches in this field can be found in [17].

2.3.3 Lot-sizing Problems

As mentioned earlier, one of important applications can be found in lot size models with quantity discounts within production-inventory systems [36]. There are two types of quantity discount situations as follows.

1. Incremental Quantity Discounts:

   Let us consider the following incremental type discount, which charges C_0 per unit for units $1, \ldots, q_1$, C_1 per unit on units $q_1, \ldots, q_2$ etc. The total cost of Q units, $C(Q)$, when $q_j < Q \leq q_{j+1}$ can be written

   $$C(Q) = R_j + C_j(Q - q_j), \quad j = 0, 1, \ldots, m \tag{2.26}$$

   where $R_j = C(q_j)$, $R_0 = 0$, $q_0 = 0$, and $q_{m+1} = \infty$. Geometrically, the total cost of Q can be represented in the same way as in Figure 2.1.

   This cost structure can be also considered within dynamic (*multi-period*) lot size models, which can be converted into an uncapacitated piecewise linear network flow model described in Section 2.2.3. It has been, however, experienced that the dynamic programming approach [15] can solve the problem more efficiently than network flow approaches due to the recursive constraint structure of these models.

2. "All Units" Discounts: (*see* Figure 2.5)

   Let us consider another type of discounts that for given quantities $q_0 = 0$,

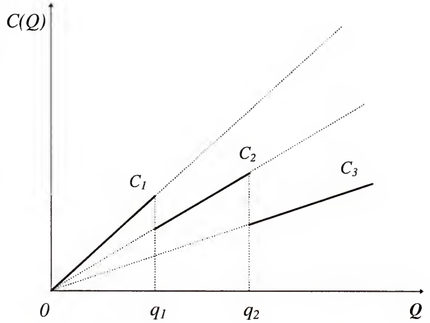

Figure 2.5. "All unit" quantity discounts.

$q_1, \ldots, q_m$, $q_j < q_{j+1}$, $j = 1, 2, \ldots, m$ and $q_{m+1} = \infty$, such that if a quantity Q is purchased, $q_j \leq Q < q_{j+1}$ then the unit cost of each of the Q units is C_j, i.e., the total cost of Q can be written as

$$C(Q) = C_j Q, \quad j = 0, 1, \ldots, m \tag{2.27}$$

where $C_j > C_{j+1}$. The cost structure of $C(Q)$ then has the form shown in Figure 2.4 (*sawtooth case*). It is referred as "all units" discounts since the discount applies to every unit purchased. Again, dynamic lot size models with this cost structure can be converted to a nonconvex piecewise linear network flow model described in Section 2.2.4.

2.4 Concluding Remarks

In this chapter, three categories of PLNFP has been identified and formulated in general formats. Some properties of problems in each category are investigated to see the insight of problems including FCNFP.

The concave PLNFP is reformulated as a FCNFP in MIP structure exploiting the concavity of arc cost functions (*see* Section 2.2.3). Moreover, the nonconvex PLNFP is also transformed to a FCNFP with a modified feasible region. This implies that the extreme point solution of the transformed FCNFP may not be an extreme point solution of the original nonconvex PLNFP.

A major benefit of the formulations introduced in this chapter is that solutions can be found by solving fixed charge problems instead of solving more difficult nonconvex optimization problems. As we can observe in the transformations to a FCNFP model, the size of FCNFP is usually quite large because of new binary variables introduced into the model. Thus, developing an efficient algorithm for large-scale FCNFPs can provide a key to solve concave and nonconvex PLNFPs in practice.

CHAPTER 3
FIXED CHARGE NETWORK FLOW PROBLEMS

3.1 Introduction

Many practical planning, transportation and distribution problems such as network design, plant location, communication, production scheduling, investment and distribution decisions can be modeled as a Fixed Charge Network Flow Problem (FCNFP).

Since this is a special subclass of the Minimum Concave Cost Network Flow Problem (MCCNFP), FCNFP also has the same characteristics as MCCNFP does. Moreover, the arc cost function in FCNFP is discontinuous at the origin. Hence, most solution approaches have utilized branch and bound techniques to find an exact solution by transforming the fixed charge problem into an equivalent 0-1 mixed integer programming problem [37]. Another classical exact solution approach is the vertex ranking procedure proposed by Murty [59], which exploits the property that a global solution occurs at the vertex of the feasible domain [64]. Due to the requirement of massive computational efforts implementing these approaches, it is still not practical to solve general large scale problems.

A number of branch and bound type methods for fixed charge transportation problems have been proposed by Barr, Glover, and Klingman [6], Cabot and Erenguc [13], Gray [31], Kennington and Unger [45], Lamar and Wallace [54], and Palekar, Karwan, and Zionts [62]. Recently, Bell, Lamar, and Wallace [8] introduced new capacity improvement techniques to enhance their branch and bound algorithm for capacitated fixed charge transportation problems. Heuristic procedures were developed by Balinski [5], Kuhn and Baumol [51], Cooper and Drebes [16], Denzler [20], Steinberg [73], Walker [79], Diaby [21], and Khang and Fujiwara [46].

Several approximate algorithms have been proposed for some specific types of problems but there is still need of a more effective solution approach to more general and larger scale problems in this class. General characteristics of some exact and approximate algorithms have been surveyed by Guisewite and Pardalos [32].

The Dynamic Slope Scaling Procedure (DSSP) described in this chapter is motivated by an economic viewpoint (marginal concept) of the fixed cost. The computational merit of this approach is that it approximates a solution for FCNFP by solving successive Linear Programming (LP) problems with recursively updated objective functions. These LP problems can be solved efficiently since the set of constraints is not changed. Accordingly, one can surely reduce the tedious computation to search the tree in branch and bound type methods by removing all the binary variables from the classical mixed integer formulation. This can be done by finding a linear factor that effectively reflects the current marginal variable cost and fixed cost at the same time depending on the current level of each variable. Thus, if the current positive level of variable can not justify the corresponding fixed cost, the DSSP updates the linear factor for better economic justification of the fixed cost. As we can see in this approach, solving a fixed charge problem can be interpreted as finding a break-even point (*level of activities*) to justify the investment of fixed costs.

A wide range of problems in the carefully designed network structure is generated and tested. The computational results show that the proposed procedure finds exact solutions in most cases of small size problems and approaches to the optimal solution in large scale problems with very small error bounds. As a challenge, test problems involving more than 10000 arcs with fixed charges are solved by the proposed algorithm and the computation times are quite reasonable. The summary of computational results is discussed in the later section.

3.2 Problem Descriptions

The FCNFP is a well-known $\mathcal{NP}$-hard problem in the subclass of minimum concave-cost network flow problems.

Consider a general capacitated fixed charge problem in the network structure. Given a directed graph $G = (N, A)$ consisting of a set N of m nodes and a set A of n arcs, then solve:

[FCNFP]

$$\min \ f(x) = \sum_{(i,j)\in A} f_{ij}(x_{ij})$$

subject to

$$\sum_{(k,i)\in A} x_{ki} - \sum_{(i,k)\in A} x_{ik} = b_i, \ \ \forall i \in N$$

and

$$0 \leq x_{ij} \leq u_{ij}, \ \ (i,j) \in A$$

where f is discontinuous at the origin and separable so that for each arc (i,j), f_{ij} has a form:

$$f_{ij}(x_{ij}) = \begin{cases} 0 & \text{if } \ x_{ij} = 0 \\ s_{ij} + c_{ij}x_{ij}, \ \ s_{ij} \geq 0 & \text{if } \ x_{ij} > 0. \end{cases}$$

To simplify the notation, we can rewrite the problem as follows:

$$\min \ f(x) = \sum_{j=1}^{n} f_j(x_j) \tag{3.1}$$

subject to

$$Ax = b \tag{3.2}$$

$$0 \leq x_j \leq u_j, \quad j = 1, 2, \ldots, n \tag{3.3}$$

where

$$f_j(x_j) = \begin{cases} 0 & \text{if } x_j = 0 \\ s_j + c_j x_j, \quad s_j \geq 0 & \text{if } x_j > 0, \end{cases} \tag{3.4}$$

A is an $m \times n$ node-arc incidence matrix and b is an m-dimensional column vector.

In many practical problems, the cost of an activity is the sum of a fixed cost and a cost proportional to the level of the activity. The FCNFP is obtained by imposing a fixed cost of $s_j \geq 0$ if there is positive flow on arc j and a variable cost c_j. Due to the discontinuity of f_j, the problem can be transformed to a 0-1 mixed integer programming problem by introducing n binary variables, indicating whether the corresponding activity is being carried out or not. Assuming $s_j > 0$, f_j can be replaced by

$$f_j = c_j x_j + s_j y_j$$

with

$$x_j \geq 0 \text{ and } y_j = \begin{cases} 0 & \text{if } x_j = 0 \\ 1 & \text{if } x_j > 0. \end{cases} \tag{3.5}$$

The above condition (3.5) can be incorporated into the capacity constraints to yield

$$0 \leq x_j \leq u_j y_j, \quad y_j \in \{0, 1\}.$$

Hence we obtain the following formulation of the fixed charge network flow problem:

[FCNFP]$_{\text{MIP}}$

$$\min \quad f(x, y) = \sum_{j=1}^{n} c_j x_j + s_j y_j$$

subject to

$$Ax = b \tag{3.6}$$

$$0 \leq x_j \leq u_j y_j, \quad j = 1, 2, \ldots, n \tag{3.7}$$

$$y_j \in \{0, 1\}, \quad j = 1, 2, \ldots, n. \tag{3.8}$$

This $[\mathbf{FCNFP}]_{\mathbf{MIP}}$ can be solved by using any type of classical branch and bound algorithms that use LP relaxations [61]. These LP relaxations can be solved efficiently by existing linear network algorithms exploiting the special structure of their feasible domain [41].

In this paper, however, we solve the problem by linearizing the objective function $f_j(x_j)$ in the original FCNFP formulation instead of using $[\mathbf{FCNFP}]_{\mathbf{MIP}}$ formulation.

3.3 Dynamic Slope Scaling Procedure

The initial motivation of this approach is to find a linear factor that effectively reflects the variable cost c_j and marginal fixed cost s_j simultaneously. This can be done by utilizing the following observation and effectively updating the linear factor.

For any given vector $\hat{x} = (\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n)^T$ and $c = (c_1, c_2, \ldots, c_n)^T$, the objective function shown in (3.4) can be expressed as a linear form with a linear factor as follows:

$$\begin{aligned} f(\hat{x}) = \bar{c}^T(\hat{x})\hat{x} &= \sum_{j=1}^{n} \bar{f}_j(\hat{x}_j) \\ &= \sum_{j=1}^{n} \bar{c}_j(\hat{x}_j)\hat{x}_j \end{aligned} \tag{3.9}$$

where the linear factor $\bar{c} = (\bar{c}_1(\hat{x}_1), \bar{c}_2(\hat{x}_2), \ldots, \bar{c}_n(\hat{x}_n))^T$ is defined by:

$$\bar{c}_j(\hat{x}_j) = \begin{cases} c_j + \frac{s_j}{\hat{x}_j} & \text{if } \hat{x}_j > 0 \\ M & \text{if } \hat{x}_j = 0. \end{cases} \tag{3.10}$$

Notice that $\bar{c}_j(\hat{x}_j)$ is the slope of the line connecting the origin and $(\hat{x}_j, f_j(\hat{x}_j))$ as shown in Figure 3.1. We denote $\bar{c}(\hat{x})$ and $\bar{c}_j(\hat{x}_j)$ as $\bar{c}$ and $\bar{c}_j$, respectively. As we can see, if $\hat{x}_j = 0$, $\bar{c}_j$ is set to be M which can be any large number, and still Eq. (3.9) holds true. However, it must be set with a certain scheme to force the proposed dynamic slope scaling procedure to find a solution. Two empirical schemes are discussed in Section 3.3.2.

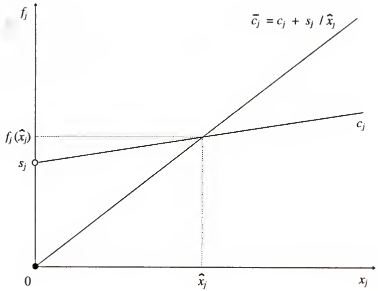

Figure 3.1. Dynamic slope scaling factor $\bar{c}_j$.

The motivation of the above linearization approach is the existence of an LP problem that has the same optimal solution as FCNFP's within the same feasible domain. It can be shown that such an LP problem exists by defining an appropriate one-to-one mapping between $\bar{c}_j$ and $\hat{x}_j$, $1 \leq j \leq n$, such as given in (3.10).

To see this at the optimum, suppose that $x^* = (x_1^*, x_2^*, \ldots, x_n^*)^T$ is an optimal solution for the FCNFP described in (3.1)-(3.4) and let $\bar{x} = (\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n)^T$ be an optimal solution for an LP problem. Then there exists an LP problem such that:

$$x_j^* = \bar{x}_j, \quad \forall j \tag{3.11}$$

and

$$f_{\mathrm{FCNFP}}(x^*) = \bar{f}_{\mathrm{LP}}(\bar{x}). \tag{3.12}$$

Now consider a recursive procedure using the above linearization approach described in (3.9) and (3.10) as follows. For each iteration $k = 1, 2, \ldots$, solve the following LP problem to find a solution, $\bar{x}^k$:

[FCNFP]$_{\mathbf{LP}}$

$$\min \quad \bar{f}^k(x) = \sum_{j=1}^{n} \bar{c}_j^k x_j \tag{3.13}$$

subject to the same constraints in (3.2) and (3.3), where the dynamic slope scaling factor is given by:

$$\begin{aligned} \bar{c}_j^k = \frac{f_j(\bar{x}_j^{k-1})}{\bar{x}_j^{k-1}} &= \frac{c_j \bar{x}_j^{k-1} + s_j}{\bar{x}_j^{k-1}} \\ &= c_j + \frac{s_j}{\bar{x}_j^{k-1}}, \quad \text{if } \bar{x}_j^{k-1} > 0. \end{aligned}$$

After finding a solution, $\bar{x}^k$ in iteration k, the current dynamic slope scaling factor is updated to be used in the next iteration with the current solution as follows:

$$\begin{aligned} \bar{c}_j^{k+1} = \frac{f_j(\bar{x}_j^{k})}{\bar{x}_j^{k}} &= \frac{c_j \bar{x}_j^{k} + s_j}{\bar{x}_j^{k}} \\ &= c_j + \frac{s_j}{\bar{x}_j^{k}}, \quad \text{if } \bar{x}_j^{k} > 0. \end{aligned} \tag{3.14}$$

In the component of the dynamic slope scaling factor in (3.14), it can be seen that $\frac{s_j}{\bar{x}_j^k}$ represents a marginal fixed cost corresponding to the current level of activity, $\bar{x}_j^k$. Thus, if the current solution of **[FCNFP]$_{\mathbf{LP}}$**, $\bar{x}_j^k$, is not enough to justify the associated fixed cost, then the next solution, $\bar{x}_j^{k+1}$, would be increased to reduce the marginal fixed cost so that the investment of the fixed cost can be justified. However, if it is impossible within the feasible region, then it will drop to zero unless the feasibility is violated. As a result, the dynamic scaling factor, $\bar{c}^k$ plays a key role in the procedure so that the current LP solution, $\bar{x}^k$ can approach to an optimal solution for the orginal FCNFP.

In the above dynamic slope scaling procedure, there are three decisions to be made:

1. how to set $\bar{c}_j^0$ to find an initial solution $\bar{x}_j^0$.
2. how to update $\bar{c}_j^{k+1}$ when $\bar{x}_j^k = 0$.
3. stopping criterion.

The strategies for these three decisions are very important because the performance of the DSSP is determined by the effectiveness of these decisions. Alternatively, the DSSP can be viewed as parametric analysis of LP (see [60]) for changing coefficients in the objective function within some ranges. In the DSSP, the range of each dynamic slope scaling factor which is a coefficient of the separable objective function in FCNFP$_{LP}$ is as follows:

$$c_j + \frac{s_j}{u_j} \le \bar{c}_j < \infty. \tag{3.15}$$

Thus, assuming that effective strategies are employed, the DSSP can produce a sequence of LP solutions (extreme points) that converge or pass through the optimal solution with a certain search-path depending on the initial solution and updating scheme.

3.3.1 Initial Solutions

As mentioned above, the efficiency of a search-path in the DSSP associated with an initial solution produced by a certain scheme may vary with problem structure. The different initial solutions can be generated by solving **[FCNFP]$_{\mathbf{LP}}$** with different initial coefficients, $\bar{c}_j^0$. Two types of candidates for initial solutions are considered as follows:

TYPE I: $\bar{x}_j^0$ with

$$\bar{c}_j^0 = c_j, \quad \text{for } j = 1, 2, \ldots, n \tag{3.16}$$

TYPE II: $\bar{x}_j^0$ with

$$\bar{c}_j^0 = c_j + \frac{s_j}{u_j}, \quad \text{for } j = 1, 2, \ldots, n \tag{3.17}$$

The base of the idea in **TYPE I** scheme is to solve the original FCNFP to find an initial solution without regarding fixed costs. The initial solution produced by **TYPE I** scheme can be far from the optimal solution at the beginning but it adjusts itself quickly once the fixed costs are involved in the next iteration of DSSP. The use of initial solutions of **TYPE I** has an advantage of a fast approach to the solution when the order of fixed costs, s_j has a similar order of unit costs, c_j. Otherwise, **TYPE II** scheme is more reliable to produce the closer initial solution for the problem. As expected, **TYPE II** scheme works slightly better for the general large scale problems in the computational experiments, however, there is no significant difference for the small size problems.

In the computational experiments, it is observed that each initial solution produces a different search-path from a vertex to another. This provides a strong possibility that the DSSP can be easily parallelized using multiple initial solutions. It should improve the performance of the DSSP for finding a better or the best solution.

3.3.2 Updating Schemes

Starting with the initial solution, $\bar{x}_j^0$ produced by either **TYPE I** or **TYPE II** and the initial coefficients, $\bar{c}_j^0$ shown above, the main iterative procedure is executed solving the **[FCNFP]$_{\mathbf{LP}}$** with the updated dynamic slope scaling factor, $\bar{c}_j^k, k = 1, 2, \ldots$. At each iteration k, several updating schemes can be considered, however, it is found that the following two schemes are reliable and effective on most problems tested.

Updating Scheme I:

- If $k = 0$, then update:

$$\bar{c}_j^{k+1} = \begin{cases} c_j + \frac{s_j}{\bar{x}_j^k} & \text{if } \bar{x}_j^k > 0 \\ c_j & \text{if } \bar{x}_j^k = 0. \end{cases} \tag{3.18}$$

Notice that if initial solution **TYPE II** is used, $\bar{c}_j^{k+1} = c_j + \frac{s_j}{u_j}$ when $\bar{x}_j^k = 0$.

- If $k \geq 1$, then update:

$$\bar{c}_j^{k+1} = \begin{cases} c_j + \frac{s_j}{\bar{x}_j^k} & \text{if } \bar{x}_j^k > 0 \\ \max_{1 \leq l \leq k}\{\ \bar{c}_j^l \mid \bar{x}_j^{l-1} > 0\} & \text{if } \bar{x}_j^k = 0. \end{cases} \tag{3.19}$$

This updating scheme is based on the history of values in the factor, $\bar{c}_j^k$ when the corresponding solution is positive. Since the maximun value of $\bar{c}_j^l$ for $l = 1, 2, \ldots, k$ contains the information of the slope that allowed the smallest possible positive level of activity j so far, **Scheme I** in (3.19) still reserves the possibility that the current zero activity can have a positive value as needed to satisfy the constraints in the following iteration. It is very important to keep this possibility because, if $\bar{c}_j^{k+1}$ is set to ∞ or an arbitrary large number when $\bar{x}_j^k = 0$, the level of activity j may not be positive later even if it should be. Thus the proper choice of slope scaling factor, $\bar{c}_j^{k+1}$ when $\bar{x}_j^k = 0$ is the key to success in finding an optimal solution in the proposed approach.

The second updating scheme is similar to **Scheme I**, but it only considers the most recent value of slope scaling factor when $\bar{x}_j^k = 0$ to update it for the next iteration. This is based on the hypothetical assertion that the most recent information is sufficient to determine the next search-path. **Scheme II** is described as follows.

Updating Scheme II:

- If $k = 0$, then update:

$$\bar{c}_j^{k+1} = \begin{cases} c_j + \frac{s_j}{\bar{x}_j^k} & \text{if } \bar{x}_j^k > 0 \\ c_j & \text{if } \bar{x}_j^k = 0. \end{cases} \tag{3.20}$$

Notice that if initial solution **TYPE II** is used, $\bar{c}_j^{k+1} = c_j + \frac{s_j}{u_j}$ when $\bar{x}_j^k = 0$.

- If $k \geq 1$, then update:

$$\bar{c}_j^{k+1} = \begin{cases} c_j + \frac{s_j}{\bar{x}_j^k} & \text{if } \bar{x}_j^k > 0 \\ \bar{c}_j^r & \text{if } \bar{x}_j^k = 0 \end{cases} \tag{3.21}$$

where r is the index of the most recent value of the slope scaling factor when $\bar{x}_j^{r-1} > 0$.

The computational results show that both schemes work well on most problems tested, however, in the case of the problems which take large number of iterations (empirically more than about 100 iterations), **Scheme II** has a slightly better performance. This implies that the most recent information of the relative solution structure is quite effective on updating the dynamic slope scaling factors.

3.3.3 Stopping Criterion

Observing the updating scheme **I** or **II** for the dynamic slope scaling factor, the stopping criterion for the procedure is determined in a natural way. Suppose that, at any iteration k,

$$\bar{x}_j^{k-1} = \bar{x}_j^k \quad \text{for all } j. \tag{3.22}$$

This means that any consecutive solutions of $[\mathbf{FCNFP}]_{\mathbf{LP}}$ are exactly the same. Then, it is clear to see that:

$$\bar{c}_j^k = \bar{c}_j^{k+1} \quad \text{for all } j. \tag{3.23}$$

Since the solution in $[\mathbf{FCNFP}]_{\mathbf{LP}}$ is determined by $\bar{c}_j$, we also must have that in iteration $k+1$,

$$\bar{x}_j^k = \bar{x}_j^{k+1} \quad \text{for all } j. \tag{3.24}$$

Accordingly, we have the situation that:

$$\begin{aligned} \bar{f}_{\mathrm{LP}}^k(\bar{x}^k) &= \bar{f}_{\mathrm{LP}}^{k+1}(\bar{x}^{k+1}) \\ &= f_{\mathrm{FCNFP}}(\bar{x}^{k+1}) \\ &= f_{\mathrm{FCNFP}}(\bar{x}^k). \end{aligned} \tag{3.25}$$

It is easy to see that once the condition (3.22) is satisfied, then there can be no more improvement. Thus, the procedure is terminated by condition (3.22).

After termination, there can be two cases in determining the best solution for the global optimality. The first case is that the procedure produces the best solution at the last iteration. The second case is that the best solution is produced before the stopping criterion is satisfied. In both cases, the best solution can be identified by updating the minimum value in each iteration until the procedure is terminated.

3.4 Computational Experiments

To show the efficiency of the proposed DSSP, we report some computational results. In addition, we evaluate relative errors to check the quality of the obtained solution. The worst relative error in our experiments is 0.65 % for the largest problem for which the exact solution is obtained by a branch and bound method.

We have solved a number of randomly generated capacitated problems of various sizes, from 12 nodes and 35 arcs up to 202 nodes and 10200 arcs. Both algorithms have been implemented in C with CPLEX 4.0 (callable library) and executed on a SUN workstation running UNIX.

Among the four different combinations of initial solution and updating schemes the combinations of **I** & **II** and **II** & **I** have a quite similar behavior to the combination of **I** & **I**, so we summarize the results in two categories. In addition, due to the limitations of memory and capability in CPLEX the exact solutions for the problems with more than 350 arcs are not available in most instances. Thus, the relative errors, i.e.,

$$\mathsf{RE} = \left[\frac{f_{\text{DSSP}} - f_{\text{exact}}}{f_{\text{exact}}}\right] \times 100$$

are shown for the problems only with up to 335 arcs. These results are summarized in Table 3.1.

Table 3.1. Summary of computational results.

Size		Scheme I & I[1]		B & B	Scheme II & II[2]	
Nodes	Arcs	RE[3](%) (min, max)	CPU (sec) (min, max)	CPU (sec) (min, max)	RE (%) (min, max)	CPU (sec) (min, max)
12	35	0.0018 (0.00, 0.02)	0.012 (0.01, 0.02)	0.022 (0.01, 0.03)	0.0018 (0.00, 0.02)	0.011 (0.01, 0.02)
18	80	0.075 (0.00, 0.18)	0.015 (0.01, 0.02)	0.180 (0.05, 0.43)	0.075 (0.00, 0.18)	0.012 (0.01, 0.02)
27	175	0.197 (0.01, 0.43)	0.037 (0.01, 0.07)	13.530 (0.94, 50.45)	0.195 (0.01, 0.39)	0.036 (0.01, 0.08)
32	225	0.352 (0.17, 0.55)	0.077 (0.04, 0.11)	241.296 (11.66, 799.69)	0.321 (0.10, 0.50)	0.063 (0.04, 0.10)
37	335	0.410 (0.12, 0.65)	0.130 (0.05, 0.21)	538.50 (81.72, 817.54)	0.344 (0.01, 0.61)	0.087 (0.03, 0.19)

1. The combination of initial solution scheme I and updating scheme I.
2. The combination of initial solution scheme II and updating scheme II.
3. The average Relative Error = $[(f_{\text{DSSP}} - f_{\text{exact}})/f_{\text{exact}}] \times 100$ (%).

In each size 10 problems have been solved by both algorithms, and the average, minimum, and maximum values of RE and CPU time (sec) are reported. The results show that the proposed algorithm finds almost exact optimal solutions within very small amount of CPU times, and provide an evidence that the combination of initial solution scheme **II** and updating scheme **II** has a slightly better performance than the other. The difference of these performances is, however, more obvious as the size of problem increases. For all the 50 test problems the overall range of relative errors (with scheme **II** and **II**, see the last column in Table 3.1) is only from 0 % to 0.61 % which is almost negligible.

To investigate the efficiency in CPU time and the extensive capability of the proposed algorithm for larger size problems, the additional computations are performed, although the exact solutions are not available. The largest problem has 10200 arcs (i.e. the 0-1 mixed integer formulation has 10200 original variables plus 10200 binary

Table 3.2. Computational results for large scale problems.

Problem	Size		Scheme I & I		Scheme II & II	
No.	Nodes	Arcs	f_{DSSP}	CPU (sec)	f_{DSSP}	CPU (sec)
1	42	440	62394	0.11	62207	0.31
2	52	650	74454	0.27	74353	0.29
3	72	1270	73521	0.99	73521	1.33
4	82	1680	72032	2.34	72032	1.90
5	102	2600	77361	4.31	76666	4.45
6	112	3110	73280	8.43	73011	7.32
7	122	3720	78754	7.94	78218	10.59
8	132	4330	80497	8.72	80381	10.40
9	142	5040	85005	18.94	84558	14.73
10	152	5750	86532	19.91	85874	33.19
11	162	6560	85687	26.47	85221	24.52
12	172	7370	86733	30.72	85640	31.19
13	182	8280	90650	31.18	90158	33.12
14	192	9190	88888	42.65	88017	66.85
15	202	10200	92134	63.01	90738	72.31

variables), and is much larger than the problems that have been considered in the literature. The CPU time for solving this problem is only about 72 seconds when the combination of initial solution scheme **II** and updating scheme **II** is used. These results are shown in Table 3.2.

As mentioned above, it is clear from Table 3.2 that the algorithm using the combination of scheme **II** and **II** produces the *better* upper bounds for all instances except the cases of problem 3 and 4, which have the exactly same objective values for both combinations. Thus, the use of this combination is recommended especially for large scale problems.

Due to the limitation of memory on the machine used, while the branch and bound algorithm is implemented, exact solutions are not available for most problems with more than 400 arcs. Thus, instead of comparing error bounds for large scale instances, a statistical analysis for RE and CPU time is performed to investigate the

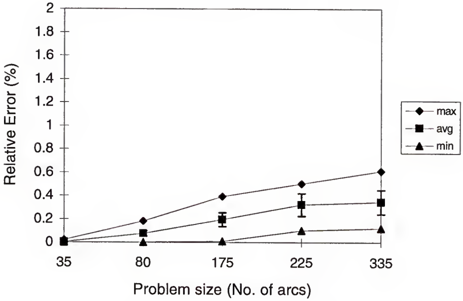

Figure 3.2. The graphical illustration of relative errors.

behavior of DSSP for those problems. Figure 3.2 shows that the regression slope of average relative errors is very small, meaning that the increasing rate of relative error is expected to be very slow as the size of problems increases. More precisely, the slope from a simple linear regression is 0.0009701. This implies that approximately 9.7% of relative error is expected when the DSSP algorithm solves a problem with 10000 arcs. Note that small vertical intervals on squared marks in Figure 3.2 represent 95% confidence intervals for the average of REs. Figure 3.3 shows a comparison of CPU time and Figure 3.4 illustrates the efficiency of DSSP algorithm in computational times for very large scale problems.

To assess the contribution of the proposed method, three additional computational experiments have been performed on the same set of test problems as follows.

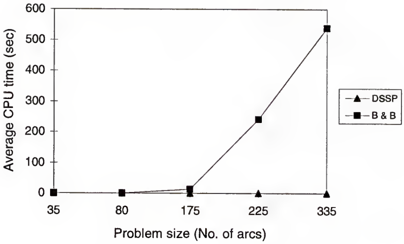

Figure 3.3. Average CPU times: DSSP vs. B & B.

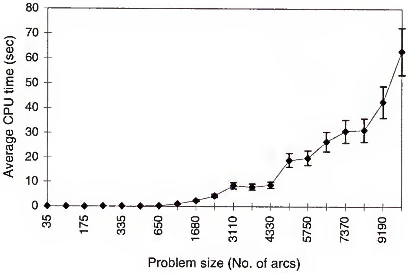

Figure 3.4. Average CPU times of DSSP for large scale problems.

Table 3.3. The Summary of $\mathsf{GAP}_{\mathrm{LP}}$, $\mathsf{RE}_{\mathrm{UB}}$ and $\mathsf{RE}_{\mathrm{ACP}}$.

Size		$\mathsf{GAP}_{\mathsf{LP}}$ (%)	$\mathsf{RE}_{\mathsf{UB}}$ (%)	$\mathsf{RE}_{\mathsf{ACP}}$ (%)
Nodes	Arcs			(min, max)
12	35	1.708	0.254	0.182 (0.00, 0.52)
18	80	3.559	0.871	0.647 (0.02, 1.27)
27	175	5.454	1.278	1.109 (0.39, 7.04)
32	225	6.867	1.804	1.541 (0.38, 8.59)
37	335	8.401	2.012	1.696 (0.99, 10.12)

Firstly, the LP relaxation subject to constraints (3.2) and (3.3) for each test problem has been solved. The values of f_{LP} for the problems in Table 3.2 are reported in Table 3.4 and the average duality gaps for the problems in Table 3.1 are summarized in Table 3.3 where

$$\mathsf{GAP}_{\mathrm{LP}} = \left[\frac{f_{\mathrm{exact}} - f_{\mathrm{LP}}}{f_{\mathrm{exact}}}\right] \times 100. \tag{3.26}$$

Secondly, an upper bound (denoted f_{UB}) is computed for each problem where f_{UB} is based on the fixed charge function in (3.4) with x_j set to the optimal flow in the LP relaxation. The values of f_{UB} for the problems in Table 3.2 are reported in Table 3.4 and the average relative errors ($\mathsf{RE}_{\mathrm{UB}}$) are reported in Table 3.3 where

$$\mathsf{RE}_{\mathrm{UB}} = \left[\frac{f_{\mathrm{UB}} - f_{\mathrm{exact}}}{f_{\mathrm{exact}}}\right] \times 100. \tag{3.27}$$

Lastly, the *Average Cost Pricing* (ACP) approach proposed by Yaged [80] has been applied to each of test problems using the initial solution TYPE II and the following updating scheme:

$$\bar{c}_j^{k+1} = \begin{cases} c_j + \frac{s_j}{\bar{x}_j^k} & \text{if } \bar{x}_j^k > 0 \\ \infty & \text{if } \bar{x}_j^k = 0. \end{cases} \tag{3.28}$$

Table 3.4. f_{LP}, f_{UB} and f_{ACP} for large scale problems.

Problem No.	Size		f_{LP}	f_{UB}	f_{ACP}
	Nodes	Arcs			
1	42	440	55963	62481	62453
2	52	650	66034	74556	74501
3	72	1270	61919	73704	73689
4	82	1680	70678	72446	72211
5	102	2600	74833	77540	77404
6	112	3110	71305	73921	73696
7	122	3720	76213	79499	78828
8	132	4330	77837	81713	81023
9	142	5040	82190	86232	85097
10	152	5750	61193	86589	86532
11	162	6560	59292	86364	85996
12	172	7370	58922	87343	87104
13	182	8280	60855	91648	90800
14	192	9190	58311	90046	89827
15	202	10200	59074	94525	92407

In the ACP algorithm, if an arc has zero flow in the current iteration, then that arc is permanently excluded from the network in subsequent iterations. Thus the ACP algorithm tends to converge quickly to a solution. However, since arcs with zero flow are never reconsidered, the ACP algorithm can get "locked-in" to a solution that is sub-optimal. The ACP objective function values (denoted f_{ACP}) are reported in Table 3.4 and the average relative errors ($\mathsf{RE}_{\mathrm{ACP}}$) including minimum and maximum values are summarized in Table 3.3 where

$$\mathsf{RE}_{\mathrm{ACP}} = \left[\frac{f_{\mathrm{ACP}} - f_{\mathrm{exact}}}{f_{\mathrm{exact}}} \right] \times 100. \tag{3.29}$$

Comparing the results from DSSP and ACP, it is clear that the DSSP improves the "locked-in" phenomenon and produces better solutions for all instances.

In addition to the above test problems, one of the well-known benchmark problems (`fixnet6`) in the literature is solved by implementing the DSSP algorithm. A brief description of this problem is given in Figure 3.5. The DSSP algorithm found

From MIPLIB 3.0:

```
* NAME:          fixnet6
* ROWS:          479
* COLUMNS:       878
* INTEGER:       378
* NONZERO:       1756
* BEST SOLN:     3983 (optimal)
* LP SOLN:       1200.88
* COMMENTS:      378 binary variables
* SOURCE:
    T. J. Van Roy (Bank Brussels Lambert, Belgium)
    Martin Savelsbergh (Eindhoven University of Technology)
```

Figure 3.5. A description of `fixnet6`.

solution 3988 in 0.25 sec (CPU time). Comparing with the exact solution (3983), the relative error is 0.126%. This result also supports the effectiveness and efficiency of the DSSP algorithm.

3.5 Local Search Procedure

In many combinatorial optimization problems, the approximate solutions from heuristic algorithms can be improved by applying various local search procedure [27, 33]. Here, we consider a greedy type local search procedure to improve the quality of the solutions obtained from the DSSP algorithm discussed previously. The computational results show that the DSSP algorithm produces the exact solutions in most small instances. However, we observe that the percentage of exact solutions found decreases as the problem size increases, although the relative error still remains small (less than 1%).

Based on the previous results that the relative errors are very small, there is a possibility that the current solution (current vertex) obtained from the DSSP is very close to the exact optimal solution (optimal vertex). In the following, we discuss a greedy adjacent vertex search scheme using the current solution vertex found in the DSSP as an initial extreme feasible solution.

3.5.1 Greedy Adjacent Vertex Local Search Algorithm

Let us define:

$$\begin{aligned} v_0 &\equiv \text{initial vertex,} \\ v_{adj} &\equiv \text{adjacent vertex of } v_0, \\ v_{best} &\equiv \text{the current best vertex,} \end{aligned}$$

$$\begin{aligned} f_0 &\equiv \text{objective function value of } v_0, \\ f_{adj} &\equiv \text{objective function value of an adjacent vertex of } v_0, \\ f_{best} &\equiv \text{the current best objective function value.} \end{aligned}$$

Now we devise the following greedy adjacent vertex local search algorithm. At each iteration of the algorithm, firstly, an adjacent feasible vertex of v_0, v_{adj} is generated and checked whether it is *better*. Secondly, If $f_1 < f_{best}$ (which is initially f_0), then update $v_{best} = v_{adj}$ and $f_{best} = f_{adj}$. Otherwise, we generate an another adjacent feasible vertex of v_0 and repeat the previous procedure until there is no more improving adjacent vertex (or the predetermined number of adjacent vertices are investigated). After completing the second step, we replace v_0 with v_{best} to reinitialize the procedure and then repeat from the first step until there is no *better* adjacent vertex found. This local search algorithm is summarized in Figure 3.6.

In order to avoid the excessive computations required by searching all vertices (global search), we may need to limit the number of iterations k, which indicates the depth of local search from the initial vertex.

3.5.2 Numerical Results

Using the current solution obtained from the DSSP algorithm as an initial feasible solution vertex, the adjacent local search algorithm described above (GAVLS) is implemented on 20 test problems for each category (total 100 problems) as used in

```
Algorithm: GAVLS
Step 0: Find an initial vertex $v_0$ (initial feasible extreme point).
          Set $v_{best} = v_0$.
          Set $f_{best} = f_0$.
          Set iteration $k = 0$.
Step 1: WHILE (there is a "better" adjacent vertex of $v_0$)
            Generate $v_{adj}$ of $v_0$.
                IF $f_{adj} < f_{best}$
                    $v_{best} \leftarrow v_{adj}$
                    $f_{best} \leftarrow f_{adj}$
                    GO TO Step 1.
                ELSE Remove $v_{adj}$.
                    GO TO Step 1.
Step 2: Reinitialization
          $v_0 \leftarrow v_{best}$
          $k \leftarrow k + 1$
          GO TO step 1.
```

Figure 3.6. A Greedy Adjacent Vertex Local Search (GAVLS) algorithm.

Table 3.1. For the all instances, the local search gradually increases the percentage of exact solution found as k increases, spending slightly more computational times. The percentage of exact solutions found (PES) and the average local search CPU times are summarized for $k = 1, 2, 3$ (see Table 3.5). The results in the first column of Table 3.5 with $k = 0$ are obtained without applying the local search procedure.

3.6 Concluding Remarks

Considering the simple and flexible structure of DSSP, the developed algorithm is quite robust so that it can be implemented to various types of problems in this class. The computational results show that the proposed solution approach (DSSP) is highly efficient and reliable. In addition, **[FCNFP]**$_{\mathbf{LP}}$ can be solved more efficiently

Table 3.5. Summary of computational results of local search.

size nodes	arcs	$k = 0$ PES[1](%)	CPU[2](s)	$k = 1$ PES	CPU	$k = 2$ PES	CPU	$k = 3$ PES	CPU
12	35	80.0	0.025	85.0	0.912	100	1.809	100	2.717
18	80	60.0	0.031	75.0	1.443	80.0	2.990	90.0	4.114
27	175	50.0	0.042	55.0	2.623	70.0	4.752	80.0	5.924
32	225	45.0	0.075	50.0	2.747	60.0	4.901	75.0	6.043
37	335	20.0	0.121	25.0	3.255	35.0	5.263	55.0	8.424

1. The percentage of exact solutions found among 20 instances for each group.
   PES(%) = [(the number of exact solutions found) / 20] × 100.
2. The average CPU time for local search (sec).

by exploiting the fact that the successive $[\mathbf{FCNFP}]_{\mathbf{LP}}$ problems use the same set of constraints in the entire procedure.

Base on the promising results from the DSSP that the overall average relative errors are quite small, a greedy adjacent vertex local search algorithm is devised to improve quality of solutions from the DSSP. The results show that the local search successfully increases the number of exact solutions found.

As a useful extension of this approach, it can be used to solve the problems with concave piecewise linear or nonconvex piecewise linear (e.g. quantity discounts) objective functions [64].

CHAPTER 4
CONCAVE PIECEWISE LINEAR NETWORK FLOW PROBLEMS

4.1 Introduction

In most network models considered in the literature, the involved costs are assumed to be linear and/or convex. In many of the applications encountered in practice however, this assumption fails to hold. In fact many of the problems arising in transportation, production-inventory, network design, location, and economic lot sizing, exhibit concave piecewise linear or economies of scale.

Piecewise Linear Network Flow Problems (PLNFP) have been studied in many different schemes for last three decades. Especially the concave PLNFP has been received a lot of attentions as concave minimization problems became a main issue in the global optimization area.

The concave PLNFP models arise in diverse applications along with a price discounting situation such as transportation, communication network design, production planning, location decision, and economic lot sizing. Another usage of PLNFP model can be to approximate a global solution for continuous smooth concave minimization problems in the branch and bound scheme.

In this chapter, three different approaches incorporating with the DSSP algorithm are discussed. The first approach is to solve a concave PLNFP by implementing a revised DSSP algorithm on the original formulation of concave PLNFP, **[CCPLNFP]**. The second approach is to implement the original DSSP algorithm for the fixed charge version of concave PLNFPs in the *extended network* described in Chapter 2. Note that usually we need to solve much larger fixed charge network flow problem than original concave PLNFP due to the separation of arcs. For instances, if each piecewise linear arc cost function in the original concave PLNFP has p linear pieces (*segments*), then the resulting fixed charge version of concave PLNFP $[\mathbf{CCPLNFP}]_{\mathbf{FC}}$ is p times larger than the original problem. This means that the DSSP algorithm can have less capability to solve large scale concave PLNFPs. To improve the performance of DSSP algorithm in the second approach, a revised DSSP algorithm incorporating with a local search scheme, so-called "Trust Intervals" and a searching strategy, a so-called "Solution Tendency" is proposed in the third approach. Computational results show that these schemes effectively remove unnecessary separated arcs to be considered from the extended network, so that the size of $[\mathbf{CCPLNFP}]_{\mathbf{FC}}$ can be reduced at each iteration of the DSSP algorithm. The computational results of implementing these algorithms developed in three different approaches are reported.

4.2 Problems and Formulations

We consider the following network problem with concave piecewise linear arc costs. Given a directed graph $G = (N, A)$ consisting of a set N of m nodes and a set A of n arcs, then solve

[CCPLNFP]

$$\min \ f(x) = \sum_{(i,j)\in A} f_{ij}(x_{ij})$$

subject to

$$\sum_{(k,i)\in A} x_{ki} - \sum_{(i,k)\in A} x_{ik} = b_i, \ \ \forall i \in N \tag{4.1}$$

$$0 \leq x_{ij} \leq u_{ij}, \ \ \forall (i,j) \in A \tag{4.2}$$

where f is separable and each f_{ij} is piecewise linear. For instance, the arc cost $f_{ij}(x_{ij})$ can be defined in a standard form as follows:

$$f_{ij}(x_{ij}) = \begin{cases} c_{ij,1}x_{ij} + s_{ij,1} & \text{if } 0 \leq x_{ij} < \lambda_{ij}^1 \\ c_{ij,2}x_{ij} + s_{ij,2} & \text{if } \lambda_{ij}^1 \leq x_{ij} < \lambda_{ij}^2 \\ \vdots & \vdots \\ c_{ij,r_{ij}}x_{ij} + s_{ij,r_{ij}} & \text{if } \lambda_{ij}^{r_{ij}-1} \leq x_{ij} \leq \lambda_{ij}^{r_{ij}} (= u_{ij}), \end{cases} \tag{4.3}$$

where λ_{ij}^r for $r = 1, 2, \ldots, r_{ij} - 1$ are breakpoints in the given interval $[0,\ u_{ij}]$ and r_{ij} is the number of linear pieces (segments) in each arc cost function f_{ij}. The problem is uncapacitated if $u_{ij} = \infty$, $\forall (i,j) \in A$. Note that due to the concavity of arc costs and a practical reason, slopes of each piecewise linear arc cost function $c_{ij,r}$, $r = 1, 2, \ldots, r_{ij}$ hold the following decreasing property:

$$c_{ij,1} > c_{ij,2} > \ldots > c_{ij,r_{ij}} > 0, \tag{4.4}$$

Based on the Arc Separation Procedure (*see* Figure 2.2) and Theorem 2.1, the original concave PLNFP can be transformed into a fixed charge model in an extended network. This extended network is denoted by $G_e(N, A_e)$ where the number of arcs $|A_e|$ is given by

$$n_e = |A_e| = \sum_{(i,j)\in A} r_{ij}.$$

Thus, the size of extended networks depends on the number of linear pieces in each arc cost function.

To formulate the fixed charge version of concave PLNFP, let

$(i,j)_r$	r-th separated arc between node i and j on an extended network;
$x_{ij,r}$	flow in the r-th separated arc between node i and j;
$f_{ij,r}$	fixed charge cost function for r-th separated arc between node i and j, i.e.

$$f_{ij,r} = c_{ij,r}x_{ij,r} + s_{ij,r} \quad \text{for } r = 1, 2, \ldots, r_{ij}.$$

Then the problem is formulated as follows:

[CCPLNFP]$_{\mathbf{FC}}$

$$\min \quad f(x) = \sum_{(i,j)\in A} \sum_{r=1}^{r_{ij}} f_{ij,r}(x_{ij,r})$$

subject to

$$\sum_{(k,i)\in A} \sum_{r=1}^{r_{ij}} x_{ki,r} - \sum_{(i,k)\in A} \sum_{r=1}^{r_{ij}} x_{ik,r} = b_i, \quad \forall i \in N \tag{4.5}$$

$$\sum_{r=1}^{r_{ij}} x_{ij,r} = x_{ij}, \qquad \forall (i,j) \in A$$

$$x_{ij,r} \geq 0, \qquad \forall (i,j) \in A,\ r = 1, 2, \ldots, r_{ij}$$

$$0 \leq x_{ij} \leq u_{ij}, \qquad \forall (i,j) \in A.$$

In the formulation, it is clear that any other set of constraints is not necessary to specify capacities for separated arcs $(i,j)_r \in A_e, \ \forall\ i,j \in N$ due to Theorem 2.1. As a result, the solution of concave PLNFP can be found by solving the fixed charge network problem formulated as above. Note that **[CCPLNFP]$_{\mathbf{FC}}$** is much larger than **[CCPLNFP]** although it is relatively easier to solve. However, the computational effort can be reduced by incorporating special search schemes with the DSSP algorithm.

4.3 Trust Intervals and Solution Tendency

Based on Theorem 2.1, we know that the optimal flow x_{ij}^* takes only one among the separated arcs between node i and j in an extended network. This provides an idea that the solution can be obtained by only considering the most trustable separated arcs in which the solution flow occurs. In other words, solving a reduced **[CCPLNFP]$_{\mathbf{FC}}$** which has much less decision variables than the original **[CCPLNFP]$_{\mathbf{FC}}$** is sufficient to find solutions. These trustable arcs can be chosen in the DSSP by identifying Trust Intervals with solution tendency for each concave piecewise linear arc cost function. Three types of solution tendency can be observed such as; 1) staying in the current interval, 2) moving down to the next lower interval, and 3) moving up to the next higher interval.

To explain such a mechanism, let us consider the k-th iteration in the DSSP. It is clear that there can be four situations in which the current solution, $\bar{x}_{ij}^k$ occurs. For each case, let us define I_{ij} to be an index set of trust intervals for arc $(i,j) \in A$.

Case I If $0 \le \bar{x}_{ij}^k < \lambda_{ij}^1$, then $I_{ij} = \{1\}$.

Case II If $\lambda_{ij}^{p-1} < \bar{x}_{ij}^k < \lambda_{ij}^p$, $p = 1, 2, \ldots, r_{ij}$, then $I_{ij} = \{p\}$.

Case III If $\bar{x}_{ij}^k = \lambda_{ij}^p$, $p = 1, 2, \ldots, r_{ij} - 1$, then $I_{ij} = \{p, p+1\}$.

Case IV If $\bar{x}_{ij}^k = \lambda_{ij}^{r_{ij}} (= u_{ij})$, then $I_{ij} = \{r_{ij}\}$.

Thus, we solve the following reduced $[\mathbf{CCPLNFP}]_{\mathbf{FC}}$ at each iteration of DSSP instead of the original $[\mathbf{CCPLNFP}]_{\mathbf{FC}}$.

$\mathbf{R} - [\mathbf{CCPLNFP}]_{\mathbf{FC}}$

$$\min \quad f(x) = \sum_{(i,j)\in A} \sum_{r \in I_{ij}} f_{ij,r}(x_{ij,r})$$

subject to

$$\sum_{(k,i)\in A} \sum_{r\in I_{ij}} x_{ki,r} - \sum_{(i,k)\in A} \sum_{r\in I_{ij}} x_{ik,r} = b_i, \quad \forall i \in N \tag{4.6}$$

$$\sum_{r\in I_{ij}} x_{ij,r} = x_{ij}, \qquad \forall (i,j) \in A$$

$$x_{ij,r} \ge 0, \qquad \forall (i,j) \in A,\ \forall r \in I_{ij}$$

$$0 \le x_{ij} \le u_{ij}, \qquad \forall (i,j) \in A.$$

A graphical example of trust intervals and solution tendency discussed above is shown in Figure 4.1.

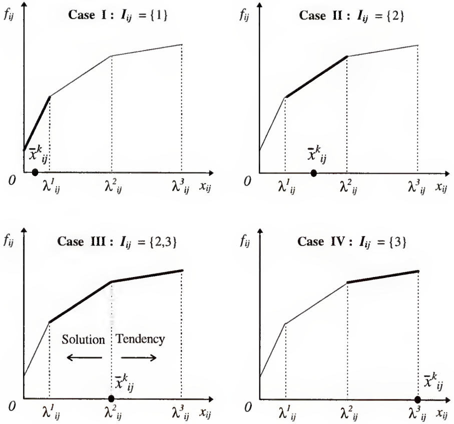
Case I : $I_{ij} = \{1\}$
Case II : $I_{ij} = \{2\}$
Case III : $I_{ij} = \{2,3\}$
Case IV : $I_{ij} = \{3\}$
f_{ij}
x_{ij}
0
λ^1_{ij}
λ^2_{ij}
λ^3_{ij}
$\bar{x}^k_{ij}$
Solution
Tendency

Figure 4.1. A graphical example of trust intervals and solution tendency

As mentioned earlier, this procedure can be considered as a local search scheme to reduce the number of feasible combinations of separated arcs investigated at each iteration of the DSSP algorithm. The DSSP algorithm incorporating with these schemes is described in the following section.

4.4 Algorithms

The DSSP algorithms with three different approaches for solving concave PLNFPs are developed and implemented on the same set of 200 test problems. Each algorithm is a variant of the original DSSP for solving the FCNFP described in Chapter 3.

Thus, the basic structure of main procedures is similar with the DSSP for FCNFPs except employing the additional search procedure with trust intervals and solution tendency. In the first approach, the DSSP algorithm for solving the original concave PLNFP model, **[CCPLNFP]** discussed in Section 4.2, we can see a potential capability of the DSSP that it can handle not only fixed charge problems but also a concave piecewise linear network problems. The performance of DSSP algorithm on the original concave PLNFP without transforming it into a $[\mathbf{CCPLNFP}]_{\mathbf{FC}}$ is shown in the computational results including a comparison with the others. Three different approaches using revised DSSP algorithms for each formulation model are described as follows.

1. **Approach I** : DSSP algorithm for solving **[CCPLNFP]**

   In this version, due to the piecewise linearity of arc costs, the initial solution and updating schemes are slightly revised.

   - Initial solution $\bar{x}_{ij}^0$ by solving LP with

     $$\bar{c}_{ij}^0 = c_{ij,r_{ij}} + \frac{s_{ij,r_{ij}}}{u_{ij}}, \quad \text{for } (i,j) \in A. \tag{4.7}$$

     It is clear to see that this initial solution is obtained by solving an LP problem with a simple linear underestimation of each concave piecewise linear arc cost. This initial solution scheme is illustrated in Figure 4.2.

   - Updating scheme:

     - If $k = 0$ then update:

     $$\bar{c}_{ij}^{k+1} = \begin{cases} c_{ij,r} + \frac{s_{ij,r}}{\bar{x}_{ij}^k} & \text{if } \lambda_{ij}^{r-1} < \bar{x}_{ij}^k \leq \lambda_{ij}^r \text{ for } r = 1, 2, \ldots, r_{ij} \\ c_{ij,r_{ij}} + \frac{s_{ij,r_{ij}}}{u_{ij}} & \text{if } \bar{x}_{ij}^k = 0. \end{cases} \tag{4.8}$$

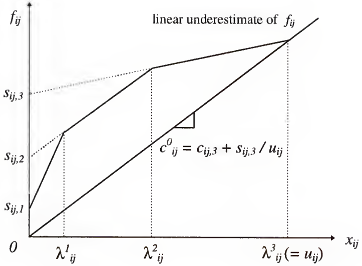
f_{ij}
linear underestimate of f_{ij}
$s_{ij,3}$
$s_{ij,2}$
$s_{ij,1}$
$c^0_{ij} = c_{ij,3} + s_{ij,3} / u_{ij}$
0
λ^1_{ij}
λ^2_{ij}
$\lambda^3_{ij} (= u_{ij})$
x_{ij}

Figure 4.2. A graphical illustration of the initial solution.

- If $k \geq 1$, then update:

$$\bar{c}_{ij}^{k+1} = \begin{cases} c_{ij,r} + \frac{s_{ij,r}}{\bar{x}_{ij}^k} & \text{if } \lambda_{ij}^{r-1} < \bar{x}_{ij}^k \leq \lambda_{ij}^r \text{ for } r = 1, 2, \ldots, r_{ij} \\ \bar{c}_{ij}^p & \text{if } \bar{x}_{ij}^k = 0 \end{cases} \tag{4.9}$$

where p is the index of the most recent value of the slope scaling factor, $\bar{c}_{ij}^k$ when $\bar{x}_{ij}^{p-1} > 0$. This updating scheme is shown in Figure 4.3.

2. **Approach II** : DSSP algorithm for solving $[\mathbf{CCPLNFP}]_{\mathbf{FC}}$

   In this approach, since the concave PLNFP has been transformed into a fixed charge problem in an extended network, the original DSSP algorithm used for FCNFP can be implemented in the same scheme. However, we need to solve very large $[\mathbf{CCPLNFP}]_{\mathbf{FC}}$ since the size of extended networks considered is much larger than the original network. Note that the number of arcs $|A_e|$ in the extended network $G_e(N, A_e)$ is given by

$$|A_e| = \sum_{(i,j) \in A} r_{ij}.$$

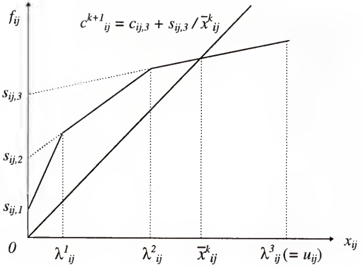

Figure 4.3. A graphical illustration of the updating scheme.

where r_{ij} is the number of linear pieces in each piecewise linear arc cost function. It is easy to see that the increasing rate of the size depends on r_{ij}. Consequently, the computational time can be increased quickly if there are many linear pieces in each arc cost.

3. **Approach III** : DSSP algorithm for solving $\mathbf{R} - [\mathbf{CCPLNFP}]_{\mathbf{FC}}$

   Since this approach is basically the same as the second approach, the same initial solutions and updating scheme as in the original DSSP for FCNFP are used in the DSSP algorithm for solving the $\mathbf{R} - [\mathbf{CCPLNFP}]_{\mathbf{FC}}$ model. However, due to employing the idea of trust intervals and solution tendency, most of unnecessary variables (arcs) to be considered at each iteration of the algorithm can be removed from the $[\mathbf{CCPLNFP}]_{\mathbf{FC}}$ model (extended network). This can be done at each iteration by identifying an index set of trust intervals I_{ij} for each $(i, j) \in A$. Thus the size of the resulting $\mathbf{R} - [\mathbf{CCPLNFP}]_{\mathbf{FC}}$ much smaller than $[\mathbf{CCPLNFP}]_{\mathbf{FC}}$ model. More precisely, the size of $\mathbf{R}-[\mathbf{CCPLNFP}]_{\mathbf{FC}}$ depends on the size of index sets I_{ij}, and can be determined at each iteration as follows. Let R be the size of $\mathbf{R}-[\mathbf{CCPLNFP}]_{\mathbf{FC}}$, then

$$R = \sum_{(i,j)\in A} |I_{ij}|$$

where $|I_{ij}|$ denotes the number of elements in I_{ij}. Note that $1 \le |I_{ij}| \le 2$. This means that the size of the problem to be solved at each iteration in this approach is at most twice of the original concave PLNFP.

In the DSSP algorithms discussed above, the same stopping criterion as shown in Chapter 3 is utilized with only small changes of index notation.

4.5 Computational Experiments

In order to evaluate the efficiency and effectiveness of the algorithms discussed in the previous section, 200 problems in different sizes and structures have been solved by applying the DSSP algorithms in three different approaches. Those test problems are divided into five groups and each group has two sets of problems, and each set contains 20 problems. The first and second sets in each group consists of problems with the fixed number of linear pieces $r_{ij} = 3$ and $r_{ij} = 5$, respectively.

The algorithms are programmed in C with CPLEX 4.0 (callable library) and implemented on a SUN workstation running UNIX. Also, the branch and bound algorithm in CPLEX 4.0 is utilized to obtain exact solutions by solving the corresponding 0-1 mixed integer programming problems. Due to the problem size and the memory limitation, exact solutions are not available when the total number of linear pieces, $\sum_{(i,j)\in A} r_{ij}$ is more than 400 in most cases. Since r_{ij} is a fixed number (3 *or* 5) in each group of problems, the total number of linear pieces can be calculated by multiplying

n and r_{ij} where n denotes the number of arcs in the original network. Relative errors are computed as follows:

$$\mathsf{RE}\ (\%) = \left[\frac{f_{\mathrm{DSSP}} - f_{\mathrm{exact}}}{f_{\mathrm{exact}}}\right] \times 100.$$

Solution times are measured by the total CPU time exclusive of I/O operations.

Computational results for **Approach I, II** and **III** are summarized in Table 4.1, 4.2 and 4.3, respectively. These results show that the DSSP in **Approach I** requires the smallest solution time (CPU time) to find solutions for all instances. However, the average relative errors in **Approach I** are bigger than those in **Approach II** and **Approach III**. This implies that the DSSP in **Approach I** takes less time to obtain solutions, but the quality of solutions from the DSSP in **Approach I** is slightly worse than the others. In problem set 4 of group P2, for example, which consists of 20 problems with 18 nodes, 80 arcs and 5 fixed number of linear pieces in each arc cost function, the average CPU time is 0.019 sec in **Approach I** whereas 0.314 sec and 0.121 sec in **Approach II** and **Approach III**, respectively. However, the average relative error is 0.713 % in **Approach I** whereas 0.411 % and 0.441 % in **Approach II** and **III**, respectively.

Comparing the results for **Approach II** and **III**, the DSSP in **Approach III** which solves problems with $\mathbf{R} - [\mathbf{CCPLNFP}]_{\mathbf{FC}}$ model turn out to be more efficient (*less solution time*) than in **Approach II**, while the quality of solutions has no significant difference. This implies that the usage of "Trust Intervals" and "Solution Tendency" incorporated with the DSSP algorithm in **Approach III** effectively removes unnecessary intervals (separated arcs) from $[\mathbf{CCPLNFP}]_{\mathbf{FC}}$ model, so that the size of problems to be solved can be significantly reduced. Especially in the case with the larger number of r_{ij}, the effectiveness of those two schemes is more significant. To see this in Table 4.2 and 4.3, the average CPU time for problem set 4 (with

Table 4.1. Computational results of DSSP in Approach I.

Problem		Size			DSSP		B & B
group	set no.	m	n	r_{ij}	RE†(%) (min, max)	CPU (sec) (min, max)	CPU (sec) (min, max)
P1	1	12	35	3	0.082 (0.00, 0.28)	0.012 (0.01, 0.02)	5.384 (1.02, 8.30)
	2	12	35	5	0.301 (0.01, 0.88)	0.014 (0.01, 0.02)	14.216 (1.29, 54.75)
P2	3	18	80	3	0.688 (0.12, 1.23)	0.016 (0.01, 0.03)	229.718 (21.03, 781.40)
	4	18	80	5	0.713 (0.13, 1.82)	0.019 (0.01, 0.04)	813.244 (45.11, 1610.38)
P3	5	27	175	3	n/a‡	0.039 (0.02, 0.07)	n/a
	6	27	175	5	n/a	0.072 (0.02, 0.08)	n/a
P4	7	32	225	3	n/a	0.048 (0.03, 0.11)	n/a
	8	32	225	5	n/a	0.121 (0.04, 0.34)	n/a
P5	9	37	335	3	n/a	0.106 (0.06, 0.32)	n/a
	10	37	335	5	n/a	0.153 (0.06, 0.45)	n/a

† The average Relative Error

‡ Exact solutions are not available from B&B due to the memory limitation.

$r_{ij} = 5$) in group P2 is reduced from 0.314 sec to 0.121 sec whereas the average CPU time is reduced from 0.087 sec to 0.080 sec in problem set 3 (with $r_{ij} = 5$). Comparisons of the average relative errors and the average CPU times in those three approaches are shown in Figure 4.4 and 4.5, respectively.

In addition, the worst relative errors in **Approach I**, **II** and **III** are 1.82 %, 0.83 % and 0.91%, respectively. As a result, the DSSP algorithms in both **Approach I** and **III** have a typical trade-off between a quality of solutions and a solution time. Thus, a decision of choosing a proper approach depends upon which attribute is more important in a given situation.

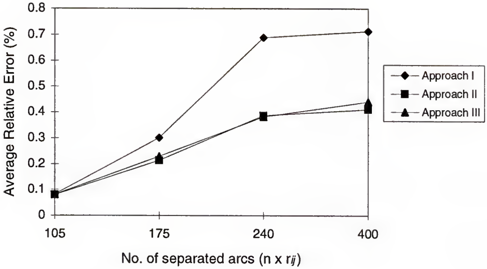

Figure 4.4. A comparison of the average relative errors.

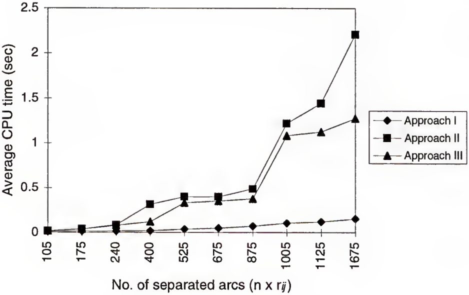

Figure 4.5. A comparison of the average CPU times.

Table 4.2. Computational results of DSSP in Approach II.

Problem		Size			DSSP		B & B
group	set no.	m	n	r_{ij}	RE[†](%) (min, max)	CPU (sec) (min, max)	CPU (sec) (min, max)
P1	1	12	35	3	0.080 (0.00, 0.18)	0.019 (0.01, 0.03)	8.914 (1.92, 19.13)
	2	12	35	5	0.214 (0.01, 0.47)	0.039 (0.01, 0.08)	16.450 (1.98, 49.17)
P2	3	18	80	3	0.387 (0.16, 0.62)	0.087 (0.03, 0.15)	204.805 (20.77, 737.61)
	4	18	80	5	0.411 (0.14, 0.83)	0.314 (0.07, 0.79)	792.133 (43.23, 1065.05)
P3	5	27	175	3	n/a	0.399 (0.07, 0.81)	n/a
	6	27	175	5	n/a	0.491 (0.09, 0.94)	n/a
P4	7	32	225	3	n/a	0.401 (0.11, 0.92)	n/a
	8	32	225	5	n/a	1.443 (0.58, 3.69)	n/a
P5	9	37	335	3	n/a	1.219 (0.49, 4.01)	n/a
	10	37	335	5	n/a	2.208 (0.92, 5.18)	n/a

† The average Relative Error

‡ Exact solutions are not available from B&B due to the memory limitation.

4.6 Concluding Remarks

In this chapter, DSSP algorithms in three different approaches have been developed and implemented for solving concave piecewise linear network flow problems. As shown by the computational results and comparisons, the DSSP in **Approach I**, which solves concave PLNFP without transforming to fixed charge network model was the most efficient in CPU time, while the quality of solutions in **Approach I** was slightly lower than those in other approaches. This results show a strong possibility

Table 4.3. Computational results of DSSP in Approach III.

Problem		Size			DSSP		B & B
group	set no.	m	n	r_{ij}	RE†(%) (min, max)	CPU (sec) (min, max)	CPU (sec) (min, max)
P1	1	12	35	3	0.081 (0.00, 0.19)	0.017 (0.01, 0.04)	8.914 (1.92, 19.13)
	2	12	35	5	0.229 (0.01, 0.59)	0.039 (0.01, 0.07)	16.450 (1.98, 49.17)
P2	3	18	80	3	0.382 (0.16, 0.62)	0.080 (0.03, 0.15)	204.805 (20.77, 737.61)
	4	18	80	5	0.441 (0.14, 0.83)	0.121 (0.05, 0.68)	792.133 (43.23, 1065.05)
P3	5	27	175	3	n/a	0.333 (0.05, 0.79)	n/a
	6	27	175	5	n/a	0.379 (0.06, 0.84)	n/a
P4	7	32	225	3	n/a	0.351 (0.09, 0.89)	n/a
	8	32	225	5	n/a	1.126 (0.27, 3.96)	n/a
P5	9	37	335	3	n/a	1.085 (0.32, 3.43)	n/a
	10	37	335	5	n/a	1.275 (0.68, 4.41)	n/a

† The average Relative Error

‡ Exact solutions are not available from B&B due to the memory limitation.

that the DSSP can be directly applied for solving more general concave and nonconvex optimization problems, such as continuous concave minimum cost network problems and nonconvex (discontinuous) piecewise linear network flow problems.

Another positive achievement in this chapter is the successful implementation of "Trust Interval" and "Solution Tendency" to reduce the size of extended networks to be considered in the DSSP algorithm. Although the DSSP algorithm in **Approach III** requires more solution time than **Approach I**, it produces better near-optimal solutions, especially in large scale problems. Considering the fact that the average CPU time in **Approach III** is still small (less than 2 seconds), this approach would be recommended in most cases except the case that the solution time is much more important than the quality of solutions. As a further research, an application of the DSSP for solving noncovex piecewise linear network problems with discontinuous arc costs (*price discounts*) so-called "staircase" and "sawtooth" arc costs is under consideration.

CHAPTER 5
NONCONVEX PIECEWISE LINEAR NETWORK FLOW PROBLEMS

5.1 Introduction

The Nonconvex Piecewise Linear Network Flow Problem (NPLNFP) is one of the most difficult problems in combinatorial network optimization [64]. In many practical applications dealing with a network structure such as transportation, logistics, supply chain management involving a production-inventory system, and facility location, we usually deal with various cost functions having *economies of scale* and/or *nonconvex piecewise linear* aspects. In most cases, these problems can be formulated as a 0-1 Mixed Integer Programming (MIP) problem which is solved by a branch and bound type method [32].

Although branch and bound algorithms can find exact solutions for small size problems, we cannot avoid exponential time and memory requirements as the problem size increases [61]. Thus, we are in need of developing efficient heuristic methods to produce "good" quality sub-optimal solutions for practical size of problems. This can be done by considering the special cost structure and solution behavior of the problem.

In particular, the problem we focus on here is to minimize a sum of discontinuous piecewise linear arc costs with a set of network constraints. This is an extension of the previous study for solving continuous concave Piecewise Linear Network Flow Problems (PLNFP) by reducing them into a fixed charge network model [50]. However, different from the concave case, there is no extreme point optimality because of the absence of concavity [34, 41]. Due to the above reason, vertex enumerating type techniques [66] can fail to find exact solutions for this problem. As a result, the solution can be attained at the boundary or interior of the feasible domain. This motivates the initial idea of adapting a domain contraction concept [75].

5.2 The Problem and Formulations

The Nonconvex Piecewise Linear Network Flow Problem (NPLNFP) can be stated as follows:

Given a directed graph $G = (N, A)$ consisting of a set N of m nodes and a set A of n arcs, then solve

[NPLNFP]

$$\min \ f(x) = \sum_{(i,j)\in A} f_{ij}(x_{ij})$$

subject to

$$\sum_{(k,i)\in A} x_{ki} - \sum_{(i,k)\in A} x_{ik} = b_i, \ \ \forall i \in N \tag{5.1}$$

$$0 \leq x_{ij} \leq u_{ij}, \ \ \forall (i,j) \in A \tag{5.2}$$

where f is separable and each f_{ij} is discontinuous piecewise linear. In general, the arc cost $f_{ij}(x_{ij})$ can be defined as follows:

$$f_{ij}(x_{ij}) = \begin{cases} c_{ij}^1 x_{ij} + s_{ij}^1, & 0 \leq x_{ij} < \lambda_{ij}^1 \\ c_{ij}^2 x_{ij} + s_{ij}^2, & \lambda_{ij}^1 \leq x_{ij} < \lambda_{ij}^2 \\ \vdots & \vdots \\ c_{ij}^{r_{ij}} x_{ij} + s_{ij}^{rij}, & \lambda_{ij}^{r_{ij}-1} \leq x_{ij} \leq \lambda_{ij}^{r_{ij}} \end{cases}$$

where λ_{ij}^r for $r = 1, 2, \ldots, r_{ij} - 1$ are breakpoints and r_{ij} denotes the number of segments in the given interval $[0, u_{ij}]$. The problem is uncapacitated if $u_{ij} = \infty$, $\forall (i,j) \in A$. Different from concave piecewise linear cases, it is assumed that each piecewise linear arc cost function is discontinuous at any breakpoint. Thus, any general concave minimization technique can not be directly implemented to solve this problem.

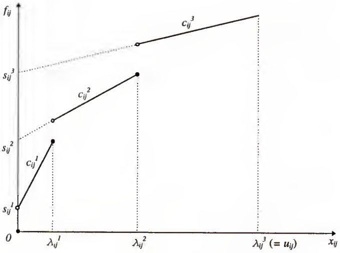

Figure 5.1. An example of staircase arc cost functions.

Nonconvex piecewise linear arc costs can be generalized in two types, so called *staircase* [11, 39, 52] and *sawtooth* [52] arc cost functions, respectively. Both arc cost functions have a very similar structure in overall shape, however, they have a different aspect at breakpoints (*see* Figure 5.1 and 5.2). It can be described in mathematical form as follows:

- "Staircase" arc cost function:

$$f_{ij}^{r-1}(\lambda_{ij}^{r-1}) < f_{ij}^{r}(\lambda_{ij}^{r-1} + \varepsilon), \quad \text{for any } \varepsilon > 0 \text{ and } r = 2, 3, \ldots, r_{ij},$$

  and

- "Sawtooth" arc cost function:

$$f_{ij}^{r-1}(\lambda_{ij}^{r-1} - \varepsilon) > f_{ij}^{r}(\lambda_{ij}^{r-1}), \quad \text{for any } \varepsilon > 0 \text{ and } r = 2, 3, \ldots, r_{ij},$$

where f_{ij}^k is an arc cost for the k-th segment in each arc cost function, f_{ij}. Moreover,

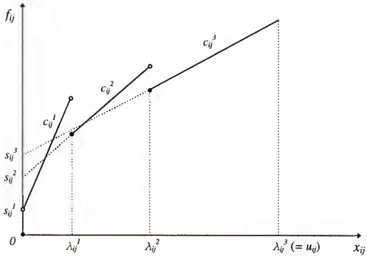

Figure 5.2. An example of sawtooth arc cost functions.

the slopes in each arc cost function, $c_{ij}^r \geq 0$, hold a decreasing property, due to the practical reason such as *quantity discounts.*

In a previous study [50], Kim and Pardalos reformulated this problem as a 0-1 Mixed Integer Programming (MIP) problem in the fixed charge structure as follows: Let us define the size of interval between adjacent breakpoints as

$$\Delta\lambda_{ij}^r = \lambda_{ij}^r - \lambda_{ij}^{r-1}, \quad \forall(i,j) \in A, \quad r = 1, 2, \ldots, r_{ij}, \tag{5.3}$$

and define the gap of function values at each breakpoint in the arc cost functions as

$$\begin{aligned} \Delta d_{ij}^r = & \ (c_{ij}^r \lambda_{ij}^{r-1} + s_{ij}^r) - (c_{ij}^{r-1}\lambda_{ij}^{r-1} + s_{ij}^{r-1}) \\ = & \ s_{ij}^r - s_{ij}^{r-1} + (c_{ij}^r - c_{ij}^{r-1})\lambda_{ij}^{r-1}, \quad \forall(i,j) \in A, \ \forall r, \end{aligned} \tag{5.4}$$

where $s_{ij}^0 = 0$ and $c_{ij}^0 = 0$ (also clearly $\Delta d_{ij}^1 = s_{ij}^1$). We now let x_{ij}^r be the part of x_{ij} that lies within level r (i.e. r-th segment), in the following sense:

$$x_{ij}^r = \begin{cases} 0 & \text{if } x_{ij} \leq \lambda_{ij}^{r-1} \\ x_{ij} - \lambda_{ij}^{r-1} & \text{if } \lambda_{ij}^{r-1} < x_{ij} \leq \lambda_{ij}^r \\ \Delta\lambda_{ij}^r & \text{if } x_{ij} \geq \lambda_{ij}^r, \end{cases} \tag{5.5}$$

and introduce new binary variables defined by

$$y_{ij}^r = \begin{cases} 1 & \text{if } \lambda_{ij}^{r-1} < x_{ij} \\ 0 & \text{otherwise.} \end{cases} \tag{5.6}$$

Then, we have

$[\mathbf{NPLNFP}]_{\mathbf{MIP}}$

$$\min \sum_{(i,j)\in A} \sum_{r=1}^{r_{ij}} (c_{ij}^r x_{ij}^r + \Delta d_{ij}^r y_{ij}^r)$$

subject to constraints in (5.1) and

$$x_{ij} = \textstyle\sum_{r=1}^{r_{ij}} x_{ij}^r, \quad \forall (i,j) \tag{5.7}$$

$$x_{ij}^r \le \Delta\lambda_{ij}^r y_{ij}^r, \quad \forall (i,j),\ r = 1,\ldots,r_{ij} \tag{5.8}$$

$$x_{ij}^{r-1} \ge \Delta\lambda_{ij}^{r-1} y_{ij}^r, \quad \forall (i,j),\ r = 2,\ldots,r_{ij} \tag{5.9}$$

$$x_{ij}^r \ge 0, \quad \forall (i,j),\ r = 1,\ldots,r_{ij} \tag{5.10}$$

$$y_{ij}^r \in \{0,1\}, \quad \forall (i,j),\ r = 1,\ldots,r_{ij}. \tag{5.11}$$

It is noticed that combining one constraint from (5.8) and one from (5.9) yields

$$\Delta\lambda_{ij}^{r-1} y_{ij}^r \le x_{ij}^{r-1} \le \Delta\lambda_{ij}^{r-1} y_{ij}^{r-1},$$

which implies

$$y_{ij}^r \le y_{ij}^{r-1}, \quad \forall (i,j),\ r > 1.$$

There is another approach to formulate the problem as a concave Minimum Cost Network Flow Problem model. In [52], Lamar described an equivalent formulation of MCNFP with general nonlinear arc costs (including the problem considered in this section) as a concave MCNFP on an extended network. The equivalence between the problems is based on converting each arc with an arbitrary cost function in the original problem into an arc with a concave piecewise linear cost function in series with a set of parallel arcs, each with a linear arc cost function (*see* [52] for details). Thus, the resulting problem is a concave MCNFP, which is different from the FCNFP formulation model shown above.

However, since both approaches are limited by problem size, a new Linear Programming (LP)-based heuristic approach is proposed by employing the *Dynamic Slope Scaling Procedure* [48] and *Domain Contraction* technique [75].

5.3 Dynamic Domain Contraction Algorithm

Consider successive LP problems which linearize the original version of NPLNFP described in Section 5.2 with linearized cost functions as follows. For each iteration k, solve the following LP problem to find a solution, $\bar{x}^k$:

$[\mathbf{NPLNFP}]^{\mathbf{k}}_{\mathbf{LP}}$

$$\min \quad \bar{f}^k(x) = \sum_{(i,j)\in A} \bar{c}^k_{ij} x_{ij} \tag{5.12}$$

subject to the same constraints in (5.1) and (5.2), where the dynamic slope scaling factor, $\bar{c}^k_{ij}$ is obtained and updated by the following scheme:

- For the initial iteration $k = 0$,

$$\bar{c}^k_{ij} = c^{r_{ij}}_{ij} + \frac{s^{r_{ij}}_{ij}}{u_{ij}} \ .$$

- For $k = 1, 2, 3, \ldots$, if the previous solution $\bar{x}^{k-1}_{ij}$ occurs in the r-th segment, then

$$\bar{c}^k_{ij} = c^r_{ij} + \frac{s^r_{ij}}{\bar{x}^{k-1}_{ij}}, \quad \text{for } \bar{x}^{k-1}_{ij} > 0.$$

The detail updating schemes when $\bar{x}^{k-1}_{ij} = 0$ can be found in [48].

Based on the Dynamic Slope Scaling Procedure (DSSP) briefly shown above, we develop a contraction rule to reduce the given feasible domain (polyhedron) for this problem. For each iteration k, the feasible domain to be considered can be reduced depending on which segments contain the current LP solution from the DSSP. If the current LP solution for arc (i,j), x_{ij}^k occurs in the r-th segment, then we can solve this problem as a fixed charge problem with an arc cost function as:

$$f_{ij}(x_{ij}) = c_{ij}^r x_{ij} + s_{ij}^r, \quad \text{for } (i,j) \in A.$$

However, due to the absence of concavity in the arc cost functions, we cannot directly decompose and reduce this problem into a simple fixed charge problem [49]. This implies that a set of constraints to specify which segment of the arc cost function to be considered must be added to the original set of constraints.

Let us define the set of additional constraints by *contraction constraints* as follows: For iteration k, if $\bar{x}_{ij}^{k-1}$ occurs in the r-th segment, then

$$\lambda_{ij}^{r-1} < x_{ij} \leq \lambda_{ij}^r, \quad \forall (i,j) \in A.$$

In addition, we define a *contraction* for iteration k by

$$[\bar{l}_{ij}^k, \bar{u}_{ij}^k], \quad \forall (i,j) \in A.$$

where $\bar{l}_{ij}^k = \lambda_{ij}^{r-1}$ and $\bar{u}_{ij}^k = \lambda_{ij}^r$ if $\bar{x}_{ij}^{k-1} \in (\lambda_{ij}^{r-1}, \lambda_{ij}^r]$. Clearly, these contraction constraints replace the original redundant capacity constraints, $0 \leq x_{ij} \leq u_{ij}$ at each iteration. This procedure is illustrated in Figure 4.3.

As a result, we solve the following recursive LP problems incorporating the Dynamic Domain Contraction (DDC) scheme described above.

$[\mathbf{NPLNFP}]_{\mathbf{LP}}^{\mathbf{k}}$ – DDC

$$\min \quad \bar{f}^k(x) = (\bar{c}^k)^T x$$

subject to

$$x \in \mathcal{D}_k = \mathcal{S} \cap \mathcal{C}_k$$

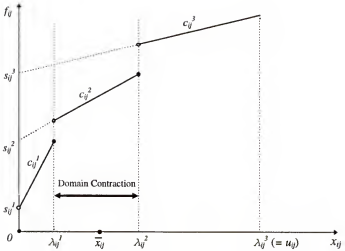

Figure 5.3. An example of domain contraction.

where

$$\mathcal{D}_k \subset \mathbb{R}^n, \text{ where } n = |A|, \tag{5.13}$$

$$\mathcal{S} = \{\, x \mid Ax = b,\ A \text{ is a node-arc incidence matrix}\} \text{ and} \tag{5.14}$$

$$\mathcal{C}_k = \{\, x \mid \bar{l}^k \leq x \leq \bar{u}^k\}. \tag{5.15}$$

In this model, we solve LP problems over the reduced domain $\mathcal{D}_k$ which is dynamically updated based on the previous LP solution $\bar{x}^{k-1}$. A series of reduced domains $\mathcal{D}_k$ may produce different types of solutions (i.e. interior point solution, boundary (not vertex) point solution, or extreme point solution) for the original problem **[NPLNFP]**, according to the relation between $\mathcal{S}$ and $\mathcal{C}_k$.

It is clear that there are two types of reduced domains $\mathcal{D}_k$ which characterize the obtained solution in this domain contraction procedure. The first case is that if $\mathcal{C}_k \subset \mathcal{S}$ which implies that $\mathcal{D}_k = \mathcal{C}_k$, then we obtain an *interior point solution* for the original nonconvex problem, although the LP solution occurs at a vertex of

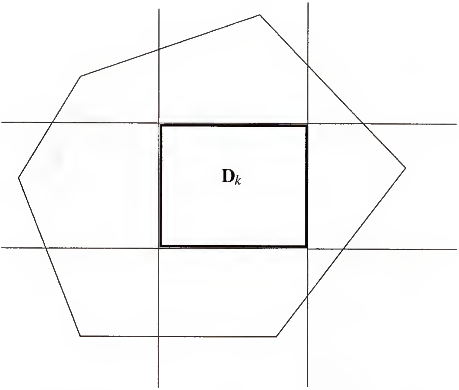

Figure 5.4. An example of the reduced domain: Case I.

$\mathcal{D}_k$ (*see* Figure 5.4). Notice that the reduced feasible domain $\mathcal{D}_k$ is a n-dimensional polytope generated by n contraction constraints in this case. The second case is that if $\mathcal{C}_k \not\subset \mathcal{S}$, so that the reduced domain $\mathcal{D}_k$ contains some boundary points and/or extreme points of $\mathcal{S}$, then we may have boundary or extreme point solution for the original problem (*see* Figure 5.5).

As a result, this dynamic domain contraction algorithm produces a series of reduced feasible domains $\mathcal{D}_k$ to search the sub-domain which contains the solution, and the algorithm terminates when there is no more improvement in successive LP solutions.

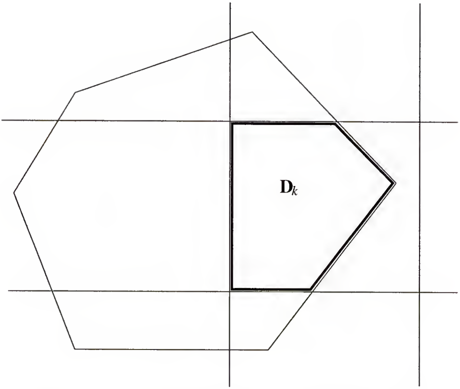

Figure 5.5. An example of the reduced domain: Case II.

5.4 Computational Experiments

Due to the difficulty of finding a set of known test problems, we generate five groups of pseudo-random problems, whose numbers of arcs are ranged from 35 to 335. Each group has two sets of problems, and each set contains 20 problems. The first set and second set in each group consist of problems with the fixed number of linear segments $r_{ij} = 3$ and $r_{ij} = 5$, respectively. Note that the actual size of problems (the number of binary variables) in the 0-1 mixed integer programming formulation, $[\mathbf{NPLNFP}]_{\mathrm{MIP}}$ is given by $|A|r_{ij}$ when r_{ij} is a constant for all arcs. This estimation is based on the extended network [52] from applying the Arc Separation Procedure (ASP) for the original network. Thus, more precisely, the total number of arcs in the extended network structure when each arc cost has a different r_{ij} is given by

$$n_e = \sum_{(i,j)\in A} r_{ij} \tag{5.16}$$

where n_e denotes the number of arcs, $|A_e|$ in the extended network, $G_e = (A_e, N)$. This means that the actual number of binary variables in the $[\mathbf{NPLNFP}]_{\mathbf{MIP}}$ formulation ranges from 105 up to 1675 in our test problems.

The algorithm consists of two parts, the domain contraction subroutine (`DDC`) to generate $\mathcal{D}_k$ and the dynamic slope scaling subroutine (`DSSP`). The algorithm is implemented in standard C with CPLEX 4.0 (callable library) and executed on a SUN workstation running UNIX. In addition, we apply the CPLEX branch and bound algorithm to the test problems for the exact solutions, in order to assess the effectiveness and efficiency of the proposed DDC algorithm. The computational results are summarized in Table 5.1.

It is observed that the branch and bound algorithm has difficulty to obtain the exact solutions for the problems having more than 80 arcs with 5 segments (i.e. $n_e = 80 \times 5 = 400$), due to the memory limitation.

For each set of 10 problems, the minimum, average and maximum relative errors, and the computational times(sec) are reported. The relative errors are computed as follows:

$$\mathsf{RE}\ (\%) = \left[\frac{f_{\mathrm{DDC}} - f_{\mathrm{exact}}}{f_{\mathrm{exact}}}\right] \times 100,$$

and the computational time is measured by the total CPU time excluding `I/O` operations. For example, the average relative error in problem set #4 is 0.628%, the average CPU time is 0.073(sec), and the maximum relative error is 0.82% from implementing the proposed DDC algorithm on 20 instances. For problem set #5 throurh #10, the average relative errors are not available since the branch and bound algorithm fails to find exact solutions for most instances.

Table 5.1. Summary of computational results for 200 problems.

Problem		Size				DDC-DSSP		B & B
Group	set no.	m	n	r_{ij}	n_e	RE[†](%) (min, max)	CPU (sec) (min, max)	CPU (sec) (min, max)
G1	1	12	35	3	105	0.102 (0.00, 0.35)	0.014 (0.01, 0.03)	6.091 (1.21, 9.13)
	2	12	35	5	175	0.391 (0.02, 0.74)	0.026 (0.01, 0.04)	43.745 (8.90, 67.25)
G2	3	18	80	3	240	0.628 (0.14, 0.82)	0.073 (0.03, 0.13)	191.547 (24.74, 560.24)
	4	18	80	5	400	0.707 (0.16, 0.94)	0.475 (0.09, 1.02)	965.355 (65.46, 1799.32)
G3	5	27	175	3	525	n/a[‡]	0.533 (0.16, 1.29)	n/a
	6	27	175	5	875	n/a	0.959 (0.47, 1.93)	n/a
G4	7	32	225	3	675	n/a	0.692 (0.29, 1.54)	n/a
	8	32	225	5	1125	n/a	1.304 (0.57, 3.80)	n/a
G5	9	37	335	3	1005	n/a	1.198 (0.59, 3.21)	n/a
	10	37	335	5	1675	n/a	1.892 (0.77, 4.90)	n/a

[†] The average Relative Error

[‡] Exact solutions are not available from B&B due to the memory limitation.

In order to see the efficiency of the proposed algorithm, we may compare the solution times between DDC and B&B algorithms. The average CPU times from implementing the DDC algorithm are ranging from 0.014(sec) up to 0.475(sec) over the test problems with up to $n_e = 400$, while the average CPU times from the branch and bound algorithm are ranging from 6.091(sec) to 965.355(sec). The maximum relative error and the maximum CPU time occur in Problem set #4 as of 0.94% and 1.02(sec), respectively, while the maximum CPU time of the branch and bound is 1799.32(sec) for the same size of test problems. The results show that the proposed solution approach can produce "high" quality near-optimal solutions efficiently for such a difficult class of nonconvex piecewise linear network flow problems described in Section 5.2, and the performance of the algorithm is stable regarding time and quality.

5.5 Concluding Remarks

Different from the traditional solution methods including branch and bound and cutting planes, the proposed DDC algorithm with `DSSP` consists of simply solving recursively updated LP problems (`DSSP`) and dynamically reducing the feasible domain (`DDC`). Thus, the implementation of the DDC algorithm is fairly simple. However, the results show that the algorithm generates high quality solutions (with less than 1% error) efficiently. In addition, due to its simple and flexible structure, the proposed method is robust for various problems in this class.

For the future work, the quality of solutions obtained by the DDC algoritm can be improved by incorporating a local search procedure.

CHAPTER 6
FUTURE RESEARCH

6.1 Application in Global Supply Chain Management

Management of the supply chain is becoming an increasingly important issue in virtually all industries. As firms seek greater market share, reduced costs and improved efficiency become critical. Logistics and supply chain management is one area in which cost reductions and strategic advantages are being sought [10].

A supply chain is the entire product delivery structure including the plant, storage warehouses, distributors, and retailers. In this global structure, there are costs associated with moving the material (*or products*) from one location to another, with building facilities for storage, and for transactions at each stage of the chain. However, the costs of holding inventories are perhaps the most significant. It has been observed that a serial supply chain structure has the tendency to amplify demand variation, resulting in much larger-than-anticipated inventories (*Bullwhip Effect*[1]) [12, 55].

In order to see the importance of inventory management, we need to discuss logistics in an integrated supply chain system. Logistics is the set of functions associated with the flow of goods, information, and payments among suppliers and consumers from the origination of raw material to final recycling or disposal of finished goods. Companies increasingly view logistics as a crucial element in their corporate strategy. If modern enterprises are to meet customer demands for responsiveness, customization, quality, and value, they must effectively and efficiently link their entire global supply chain.

---

[1] The bullwhip effect is the name given to the phenomenon of increased variation in the pattern of orders placed as one moves from the consumer level to intermediate distributors to the factory.

Logistics supply chains are the physical components including the manufacturing plants, warehouses, vehicles, and transportation infrastructure. Logistics information and decision technologies are the computer related technologies utilized in the design, planning, and operation of supply chains including monitoring status within the supply chain and communicating among supply chain elements. A totally integrated logistics value chain is seamless across all supply chain elements and optimized with regard to both service and cost.

A Network Approach

Unfortunately, due to the complexity of supply chain structures including production and inventory systems, it is very difficult to develop a global solution technique for supply chain management problems. Recently, many companies have tried to *locally* optimize their supply chain systems (e.g. component by component), and then integrate the components and implement the local solutions to their global systems. It is clear that the integrated local solutions may not provide a global solution to the entire supply chain management problem.

There is an interesting approach to handle these problems globally in various network structures. Although the approach may have some difficulties in representing very specific and complicated systems, we still have two advantages from this approach. The first advantage is that once the problem is formulated in a network structure, we can obtain a global solution by using an appropriate network optimizing technique which can handle nonconvex costs (*fixed charge and/or quantity discounts*). This network solution is also very useful to view the entire flow diagram of goods, so that one can monitor the current status of each component in the supply chain. The second advantage is more realistic: once a network structure for the problem is obtained, then it can be utilized in designing data communication networks for supply chain management.

Historically, a network approach for production-inventory problems was originally introduced by Zangwill [82], as a concave cost network flow problem. Consider the following formulation of the classical *Wagner-Whitin* model [78]. Let x_j be the amount produced in the beginning of period j and I_j be the inventory at the end of period j. Assuming the market demand in period j is given (*or estimated*) by d_j, we obtain the following formulation.

$$\min \sum_{j=1}^{n} P_j(x_j) + \sum_{j=1}^{n} H_j(I_j) \tag{6.1}$$

subject to

$$\sum_{j=1}^{n} x_j = \sum_{j=1}^{n} d_j \tag{6.2}$$

$$x_j + I_{j-1} - I_j = d_j, \qquad j = 1, 2, \ldots, n \tag{6.3}$$

$$x_j \geq 0,\ I_j \geq 0, \qquad j = 1, 2, \ldots, n \tag{6.4}$$

$$I_0 = I_n = 0 \tag{6.5}$$

where $P_j(x_j)$ is a production cost function and $H_j(I_j)$ is an inventory cost function. These costs are assumed concave, especially fixed charge and/or concave piecewise linear, in most real-world situations. Note that, clearly, Eq. 6.2 is a redundant constraint, and the above system describes a network structure. An example of the network representation of the model is shown in Figure 6.1.

Now let us consider a typical supply chain with four levels, from a manufacturing system to customers (*see* Figure 6.2). Clearly, it has a network structure. In most cases, this network problem will involve transportation and/or vehicle routing problem(s), which can be solved efficiently by an appropriate network optimization algorithm.

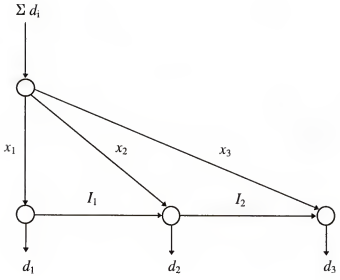

Figure 6.1. The network representation of the Wagner-Whitin model.

The main idea here is that the network model described in (6.1) - (6.5) can be embedded into the main network structure of a given supply chain. Then we have an extended network structure for the supply chain, which may have fixed charges and/or concave piecewise linear costs. As a result, the problem becomes a fixed charge or concave piecewise linear network flow problem, which can be transformed to a 0-1 mixed integer programming problem and solved by the branch and bound method [32, 41].

Due to the limitation (computing time and memory) of the branch and bound method, a number of approximation methods have been developed in this field over the last several decades. Useful information can be found in [32]. Recently, Kim and Pardalos [48] developed an efficient heuristic approach for the fixed charge network flow problem. Their computational results show that the heuristic algorithm, the

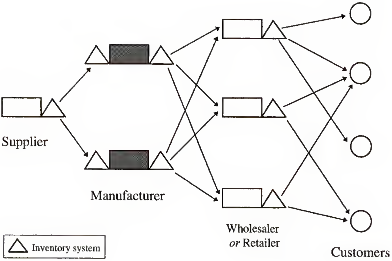

Figure 6.2. A typical supply chain model.

so-called "Dynamic Slope Scaling Procedure," provides quite efficient and effective solutions exploiting the separability of network structures.

6.2 Piecewise Concave Nonlinear Arc Costs

As a special extension of nonconvex piecewise linear arc costs, piecewise concave nonlinear arc costs can be considered for more complex, but more realistic economies of scale cost structure in various application problems. This cost structure may also include fixed charges in each segment of the arc cost function. Like piecewise linear cases, there can be two distinct cost structures such as *Continuous Piecewise Concave* (*see* Figure 6.3) and *Discontinuous Piecewise Concave* arc costs (*see* Figure 6.4). The difference between these cost structures is based on whether the fixed charge is incurred at each breakpoint or not. One of the simplest examples for these costs can be represented as a piecewise quadratic function.

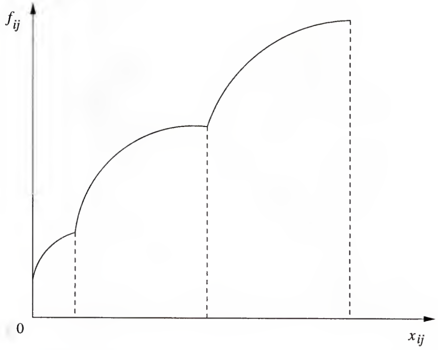

Figure 6.3. A continuous piecewise concave arc cost function.

There can be two approaches for solving the problem with these arc costs. The first approach is to directly apply the Dynamic Slope Scaling Procedure with simple modifications in the initial solution schemes discussed in Chapter 3. The second approach is to approximate these nonlinear cost function in each segment by a concave piecewise linear function, and then solve the piecewise linearized problem using the algorithms developed in Chapter 4 and 5. The latter approach would be a better way to exploit the previous development of efficient algorithms for PLNFPs. Note that, in order to generate better approximate solution, one may use more linear segments in the above piecewise linear approximation. However, the computational effort will increase as use more linear segments in the approximation procedure. This extension is one of the most realistic representations of the cost in many practical application models exhibiting multilevel economies of scale.

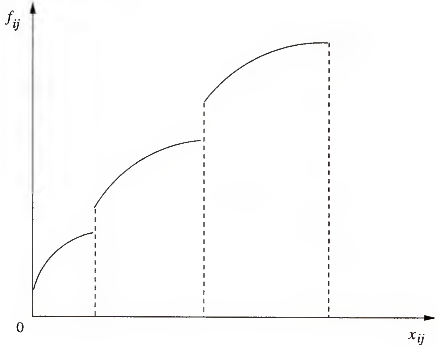

Figure 6.4. A discontinuous piecewise concave arc cost function.

REFERENCES

[1] R.K. Ahuja, T.L. Magnanti and J.B. Orlin. *Network Flows: Theory, Algorithms and Applications*, Prentice Hall, New Jersey, 1993.

[2] U. Akinc and B.M. Khumawala. An Efficient Branch and Bound Algorithm for the Capacitated Warehouse Location Problem. *Management Science* **23**, pp. 585-594, 1977.

[3] F. Archetti and F. Schoen. A Survey on the Global Optimization Problem: General Theory and Computational Approaches. *Annals of Operations Research* **1**, pp. 87-110, 1984.

[4] A. Balakrishnan and S.C. Graves. A Composite Algorithm for a Concave-Cost Network Flow Problem. *Networks* **19**, pp. 175-202, 1989.

[5] M.L. Balinski. Fixed Cost Transportation Problem. *Naval Research Logistics Quarterly* **8**, pp. 41-54, 1961.

[6] R.S. Barr, F. Glover and D. Klingman. A New Optimization Method for Large Scale Fixed Charge Transportation Problems. *Operations Research* **29**, pp. 448-463, 1981.

[7] J.E. Beasly. Lagrangean Heuristics for Location Problem. *European Journal of Operational Research* **65**, pp. 383-399, 1993.

[8] G.J. Bell, B.W. Lamar and C.A. Wallace. Capacity Improvement, Penalties, and the Fixed Charge Transportation Problem. Research Paper, Dept. of Management, Univ. of Canterbury, Christchurch, New Zealand 1996.

[9] H.P. Benson. Concave Minimization: Theory, Applications, Algorithms. In R. Horst and P.M. Pardalos (*eds.*), *Handbook of Global Optimization*, Kluwer Academic Publishers, Dordrecht, pp. 43-148, 1995.

[10] B.S. Blanchard. *Logistics Engineering and Management*, *4th* Ed., Prentice Hall, New Jersey, 1992.

[11] C.T. Bornstein and R. Rust. Minimizing a Sum of Staircase Functions under Linear Constraints. *Optimization* **19**, pp. 181-190, 1988.

[12] J.F. Burns and B.D. Sivazlian. Dynamic Analysis of Multi-Echelon Supply Systems. *Computer & Industrial Engineering* **2**, pp. 181-193, 1978.

[13] A.V. Cabot and S.S. Erenguc. Some Branch-and-Bound Procedures for Fixed-Cost Transportation Problems. *Naval Research Logistics Quarterly* **31**, pp. 145-154, 1984.

[14] H.D. Chen, D.W. Hearn and C.Y. Lee. A New Dynamic Programming Algorithm for the Single Item Capacitated Dynamic Lot Size Model. *Journal of Global Optimization* **4**, pp. 285-300, 1994a.

[15] H.D. Chen, D.W. Hearn and C.Y. Lee. A Dynamic Programming Algorithm for Dynamic Lot-Size Models with Piecewise Linear Costs. *Journal of Global Optimization* **4**, pp. 397-413, 1994b.

[16] L. Cooper and C. Drebes. An Approximate Solution Method for the Fixed Charge Problem. *Naval Research Logistics Quarterly* **14**, pp. 101-113, 1967.

[17] G. Cornuejols, R. Sridharan and J.M. Thizy. A Comparison of Heuristics and Relaxations for the Capacitated Plant Location Problem. *European Journal of Operational Research* **50**, pp. 280-297, 1991.

[18] G.B. Dantzig. Application of the Simplex Method to a Transportation Problem. In T.C. Koopmans (*ed.*), *Activity Analysis of Production and Allocation*, John Wiley, New York, 1951.

[19] P.S. Davis and T.L. Ray. A Branch-and-Bound Algorithm for the Capacitated Facilities Location Problem. *Naval Research Logistics Quarterly* **16**, pp. 331-344, 1969.

[20] D.R. Denzler. An Approximate Algorithm for the Fixed Charge Problem. *Naval Research Logistics Quarterly* **16**, pp. 411-416, 1964.

[21] M. Diaby. Successive Linear Approximation Procedure for Generalized Fixed-Charge Transportation Problem. *Journal of the Operational Research Society* **42**, pp. 991-1001, 1991.

[22] L.C.W. Dixon and G.P. Szegö. The Global Optimization Problem: An Introduction. In L.C.W. Dixon and G.P. Szegö (*eds.*), *Towards Global Optimization* **2**, North-Holland, Amsterdam, 1978.

[23] D.-Z. Du and P.M. Pardalos (*eds.*). *Network Optimization Problems: Algorithms, Complexity and Applications*, World Scientific, Singapore, 1993.

[24] M.A. Efroymson and T.L. Ray. A Branch-and-Bound Algorithm for Plant Location. *Operations Research* **14**, pp. 361-368, 1966.

[25] M. Florian and P. Robillard. An Implicit Enumeration Algorithm for The Concave Cost Network Flow Problem. *Management Science* **18**, pp. 185-193, 1971.

[26] R.L. Francis, L.F. McGinnis and J.A. White. *Facility Layout and Location: An Analytical Approach*, *2nd* Ed., Prentice Hall, New Jersey, 1992.

[27] G. Gallo and C. Sodini. Adjacent Extreme Flows and Applications to Min Concave-Cost Flow Problems. *Networks* **9**, pp. 95-121, 1979.

[28] G. Gallo, C. Sandi and C. Sodini. An algorithm for the min concave cost flow problem. *European Journal of Operations Research* **4**, pp. 248-259, 1980.

[29] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, San Francisco, 1979.

[30] A. Geoffrion and R. McBride. Lagrangean Relaxation applied to Capacitated Facility Location Problems. *AIIE Transactions* **10**, pp. 40-47, 1978.

[31] P. Gray. Exact Solution of the Fixed-Charge Transportation Problem. *Operations Research* **19**, pp. 1529-1538, 1971.

[32] G.M. Guisewite and P.M. Pardalos. Minimum Concave Cost Network Flow Problems: Applications, Complexity, and Algorithms. *Annals of Operations Research* **25**, pp. 75-100, 1990.

[33] G.M. Guisewite and P.M. Pardalos. Algorithms for the Single-Source Uncapacitated Minimum Concave Cost Network Flow Problem. *Journal of Global Optimization* **1**, pp. 245-265, 1991.

[34] G.M. Guisewite and P.M. Pardalos. Global Search Algorithms for Minimum Concave Cost Network Flow Problems. *Journal of Global Optimization* **1**, pp. 309-330, 1991.

[35] G.M. Guisewite and P.M. Pardalos. A Polynomial Time Solvable Concave Cost Network Flow Problem. *Networks* **23**, pp. 143-147, 1993.

[36] G. Hadley and T.M. Whitin. *Analysis of Inventory Systems*, Prentice Hall, New Jersey, 1963.

[37] W.M. Hirsch and G.B. Dantzig. The Fixed Charge Problem. *Naval Research Logistics Quarterly* **15**, pp. 413-424, 1968.

[38] F.L. Hitchcock. The Distribution of a Product from Several Sources to Numerous Locations. *Journal of Mathematics and Physics* **20**, pp. 224-230, 1941.

[39] K. Holmberg. Solving the staircase cost facility location problem with decomposition and piecewise linearization. *European Journal of Operational Research* **75**, pp. 41-61, 1994.

[40] R. Horst and H. Tuy. The Geometric Complementarity Problem and Transcending Stationarity in Global Optimization. DIMACS series in *Discrete Mathematics and Theoretical Computer Science* **4**, pp. 341-354, 1991.

[41] R. Horst and P.M. Pardalos (*eds.*). *Handbook of Global Optimization*, Kluwer Academic Publishers, Dordrecht, 1995.

[42] R. Horst, P.M. Pardalos and N.V. Thoai. *Introduction to Global Optimization*, Kluwer Academic Publishers, Dordrecht, 1995.

[43] P.A. Jensen, J.W. Barnes. *Network Flow Programming*, John Wiley, New York, 1995.

[44] L. Kantorovich. *Mathematical Methods in the Organization and Planning of Production*, Publication House of Leningrad University, Leningrad, 1939. Translated: *Management Science* **6**, pp. 366-422, 1960.

[45] J. Kennington and V.E. Unger. A New Branch-and-Bound Algorithm for the Fixed Charge Transportation Problem. *Management Science* **22**, pp. 1116-1126, 1976.

[46] D.B. Khang and O. Fujiwara. Approximate Solution of Capacitated Fixed-Charge Minimum Cost Network Flow Problems. *Networks* **21**, pp. 689-704, 1991.

[47] B.M. Khumawala. An Efficient Branch-and-Bound Algorithm for the Warehouse Location Problem. *Management Science* **18**, B718-B731, 1972.

[48] D. Kim and P.M. Pardalos. A Solution Approach to the Fixed Charge Network Flow Problem Using a Dynamic Slope Scaling Procedure. *Operations Research Letters*, 1999, to appear.

[49] D. Kim and P.M. Pardalos. Piecewise Linear Network Flow Problems. In C.A. Floudas and P.M. Pardalos (*eds.*), *Encyclopedia of Optimization*, Kluwer Academic Publishers, Dordrecht, 1999, to appear.

[50] D. Kim and P.M. Pardalos. Dynamic Slope Scaling and Trust Interval Techniques for Concave Piecewise Linear Network Flow Problems. ISE Department, University of Florida, Technical Report, 1999.

[51] H. Kuhn and W. Baumol. An Approximate Algorithm for the Fixed Charge Transportation Problem. *Naval Research Logistics Quarterly* **9**, pp. 1-15, 1962.

[52] B.W. Lamar. A Method for Solving Network Flow Problems with General Nonlinear Arc Costs, in D.-Z. Du and P.M. Pardalos (*eds.*), *Network Optimization Problems*, World Scientific, Singapore, 1993.

[53] B.W. Lamar. An Improved Branch and Bound Algorithm for Minimum Concave Cost Network Flow Problems. *Journal of Global Optimization* **3**, pp. 261-287, 1993.

[54] B.W. Lamar and C.A. Wallace. Revised-Modified Penalties for Fixed Charge Transportation Problems. *Management Science*, **43**, pp. 1431-1436, 1997.

[55] H.L. Lee, V. Padmanabhan and S. Whang. Information Distortion in a Supply Chain: The Bullwhip Effect. *Management Science* **43**, No. 4, pp. 546-558, 1997.

[56] T.L. Magnanti and R.T. Wong. Network Design and Transportation Planning: Models and Algorithms. *Transportation Science* **18**, pp. 1-55, 1984.

[57] O.L. Mangasarian and T.H. Shiau. A Variable Complexity Norm Maximization Problem. Computer Science Department, University of Wisconsin, Technical Report **568**, 1984.

[58] P. McKeown. A Vertex Ranking Procedure for Solving the Linear Fixed-Charge Problem. *Operations Research* **23**, pp. 1183-1191, 1975.

[59] K.G. Murty. Solving the Fixed Charge Problem by Ranking the Extreme Points. *Operations Research* **16**, pp. 268-279, 1968.

[60] K.G. Murty. *Linear and Combinatorial Programming*, John Wiley, New York, 1976.

[61] G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization*, John Wiley, New York, 1988.

[62] U.S. Palekar, M.H. Karwan and S. Zionts. A Branch-and-Bound Method for Fixed Charge Transportation Problem. *Management Science* **36**, pp. 1092-1105, 1990.

[63] P.M. Pardalos and J.B. Rosen. Methods for Global Concave Minimization: A Bibliographic Survey. *SIAM Review* **28**, pp. 367-379, 1986.

[64] P.M. Pardalos and J.B. Rosen. *Constrained Global Optimization: Algorithms and Applications*, Lecture Notes in Computer Science **268**, Springer-Verlag, Berlin, 1987.

[65] P.M. Pardalos and G. Schnitger. Checking Local Optimality in Constrained Quadratic Programming is NP-hard. *Operations Research Letters* **7**, pp. 33-35, 1987.

[66] P.M. Pardalos. Enumerative Techniques for Solving some Nonconvex Global Optimization Problems. *OR Spectrum* **10**, pp. 29-35, 1988.

[67] P.M. Pardalos and S.A. Vavasis. Quadratic Programming with One Negative Eigenvalue is NP-hard. *Journal of Global Optimization* **1**, pp. 29-35, 1990.

[68] P.M. Pardalos, D. Hearn and W. Hager (*eds.*). *Network Optimization*, Lecture Notes in Economics and Mathematical Systems **450**, Springer-Verlag, Berlin, 1997.

[69] P.M. Pardalos and D.-Z. Du (*eds.*). *Network Design: Connectivity and Facilities Location*, DIMACS Series **40**, American Mathematical Society, 1998.

[70] P. Rech and L.G. Barton. A Non-Convex Transportation Algorithm. In E.M.L. Beale (*ed.*), *Applications of Mathematical Programming Techniques*, American Elsevier Publishers, New York, 1970.

[71] G. Sa. *Concave Programming in Transportation Networks*, Ph.D. dissertation, Sloan School of Management, M.I.T., 1968.

[72] R.M. Soland. Optimal Facility Location with Concave Costs. *Operations Research* **22**, pp. 373-382, 1974.

[73] D.I. Steinberg. The Fixed Charge Problem. *Naval Research Logistics Quarterly* **17**, pp. 217-236, 1970.

[74] P.T. Thach. A Decomposition Method for the Min Concave Cost Flow Problem with a Staircase Structure. *Japan J. Appl. Math.* **7**, pp. 103-120, 1990.

[75] L.S. Thakur. Domain Contraction in Nonlinear Programming: Minimizing a Quadratic Concave Objective over a Polyhedron. *Mathematics of Operations Research* **16**, No. 2, pp. 390-407, 1991.

[76] A.A. Törn and A. Žilinskas. *Global Optimization*, Lecture Notes in Computer Science **350**, Springer-Verlag, Berlin, 1974.

[77] H. Tuy. Concave Programming under Linear constraints. *Dokl. Akad. Nauk. SSSR* **159**, pp. 32-35, 1964. Translated: *Soviet Math* **5**, pp. 1437-1440, 1964.

[78] H.M. Wagner and T.M. Whitin. Dynamic Version of the Economic Lot Size Model. *Management Science* **5**, No. 1, pp. 89-96, 1959.

[79] W.E. Walker. A Heuristic Adjacent Extreme Point Algorithm for the Fixed Charge Problem. *Management Science* **22**, pp. 587-596, 1976.

[80] B. Yaged. Minimum Cost Routing for Static Network Models. *Networks* **1**, pp. 139-172, 1971.

[81] W.I. Zangwill. Minimum Concave-Cost Flows in Certain Networks. *Management Science* **14**, pp. 429-450, 1968.

[82] W.I. Zangwill. A Backlogging Model and a Multi-Echelon Model of a Dynamic Economic Lot Size Production System - A Network Approach. *Management Science* **15**, No. 9, pp. 506-527, 1969.

BIOGRAPHICAL SKETCH

Dukwon Kim was born in Seoul, Korea, in January 3, 1964. He received his B.S. degree in mathematics from Korea University, Seoul, Korea, in February 1987, and an M.S. degree in operations research from Western Michigan University, Kalamazoo, Michigan, in April 1995.

Since August 1995, he has been a Ph.D. student in the Department of Industrial and Systems Engineering of the University of Florida to pursue his doctoral degree in operations research. His primary research is focused on combinatorial network optimization, with various applications including transportation, production-inventory control, global supply chain, and logistics.

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

P. M. Pardalos

Panos M. Pardalos, Chairman
Professor of Industrial and Systems
Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Donald W Hearn

Donald W. Hearn
Professor of Industrial and Systems
Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Boghos D. Sivazlian
Professor of Industrial and Systems
Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Harold P. Benson

Harold P. Benson
Professor of Decision and Information
Sciences

This dissertation was submitted to the Graduate Faculty of the College of Engineering and to the Graduate School and was accepted as partial fulfillment of the requirements for the degree of Doctor of Philosophy.

August 1999

M. J. Ohanian
Dean, College of Engineering

Winfred M. Phillips
Dean, Graduate School